repository.ub.ac.id

PERANCANGAN DAN IMPLEMENTASI ALGORITMA ENKRIPSI IDEA PADA PERANGKAT KRIPTOGRAFI BERBASIS FPGA

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh:
M. Adi Wijaya
NIM: 145150301111073

PROGRAM STUDI TEKNIK KOMPUTER
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

UNIVERSITAS BRAWIJAYA

PENGESAHAN

PERANCANGAN DAN IMPLEMENTASI ALGORITMA ENKRIPSI IDEA PADA PERANGKAT KRIPTOGRAFI BERBASIS FPGA

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun Oleh :
M. Adi Wijaya
NIM: 145150301111073

Skripsi ini telah diuji dan dinyatakan lulus pada
2 Agustus 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Wijaya Kurniawan, S.T, M.T	Ari Kusyanti, S.T, M.Sc
NIP: 19820125 201504 1 002	NIP: 19831228 201803 2 002

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP: 19710518 200312 1 001

repository.ub.ac.id

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 2 Agustus 2018

M. Adi Wijaya

NIM: 145150301111073

UNIVERSITAS BRAWIJAYA

KATA PENGANTAR

Puji syukur kepada Allah SWT atas segala karunia-Nya sehingga penulis dapat menyelesaikan skripsi ini yang berjudul "Perancangan dan Implementasi Algoritma Enkripsi IDEA pada Perangkat Kriptografi Berbasis FPGA".

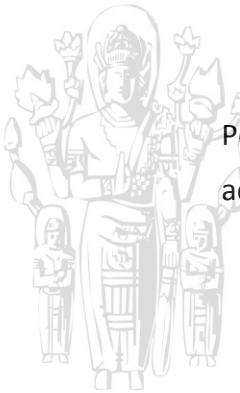
Penulisan skripsi ini digunakan bagi penulis sebagai salah satu syarat untuk memperoleh gelar sarjana Teknik pada Fakultas Ilmu Komputer Universitas Brawijaya. Dalam pengerjaan skripsi, penulis mendapatkan banyak pengalaman dan ilmu pengetahuan baru yang tidak diajarkan pada saat perkuliahan.

Dalam penulisan dan penyusunan skripsi ini tidak terlepas dari berbagai bantuan, bimbingan, dukungan serta motivasi dari berbagai pihak. Dengan anugerah Allah SWT dan dukungan dari beberapa pihak, penulis dapat melewati masa-masa sulit mengerjakan skripsi dan dapat menyelesaikan skripsi sesuai dengan target yang ditetapkan oleh penulis. Oleh karena itu, pada kesempatan kali ini penulis ingin menyampaikan ucapan terima kasih yang sebesar-besarnya kepada :

1. Bapak Wijaya Kurniawan, S.T, M.T., selaku dosen pembimbing satu dan dosen penasehat akademik yang telah membimbing penulis dari awal perkuliahan sampai akhir perkuliahan, segenap Bapak dan Ibu dosen serta karyawan yang telah mendidik dan membantu penulis selama menempuh pendidikan di Fakultas Ilmu Komputer Universitas Brawijaya.
2. Ibu Ari Kusyanti, S.T, M.Sc., selaku dosen pembimbing dua yang telah banyak membimbing, memberi pengarahan, saran, motivasi dan do'a untuk penyelesaian skripsi ini.
3. Bapak Tri Astoto Kurniawan, S.T, M.T, Ph.D selaku Ketua Jurusan Teknik Informatika Universitas Brawijaya.
4. Bapak Sabriansyah Rizqika Akbar, S.T, M.Eng., selaku Ketua Program Studi Teknik Komputer Universitas Brawijaya.
5. Bapak Eriq Muhammad Adams Jonemaro S.T, M.Kom dan Bapak Aninditya Nugroho, S.T., yang telah memberikan kesempatan dan waktunya untuk membantu penulis untuk pengadaan dan peminjaman alat laboratorium
6. Kedua orang tua penulis yaitu Alm. Gaguk Kiswoyo dan Ibu Yuliana yang selalu memotivasi, menyemangati dan mendoakan penulis serta memberikan dukungan dalam bentu moril maupun materiil demi kelancaran skripsi penulis.
7. Istania Salma selaku teman dekat penulis yang selalu memberikan motivasi dan semangat kepada penulis, menemani disaat susah dan senang, selalu ada ketika penulis membutuhkan bantuan dalam melakukan penelitian ini, selalu membuat penulis tetap maju dan menjadi lebih baik

8. Sahabat “KBMCL” yang telah memberikan semangat kepada penulis.
9. Sabahat “Go Internasional” yang selalu memacu penulis untuk terus maju dan tidak mudah menyerah, memberikan hiburan ketika penulis penat saat mengerjakan skripsi. Kepada Istania Salma, Annisa Amalia dan Falih Gozi Febrian penulis mengucapkan terima kasih.
10. Sahabat “Keluarga Dewan Perwakilan Mahasiswa 2016 dan Keluarga Dewan Perwakilan 2017” yang memberikan semangat dan motivasi di setiap saat kepada penulis yang namanya tidak bisa penulis sebutkan satu persatu penis mengucapkan terimakasih.
11. Dan seluruh teman-teman yang memberikan dukungan, motivasi dan bantuan penulis yang tidak bisa penulis sebutkan satu persatu, penulis mengucapkan terima kasih

Malang, 2 Agustus 2018



Penulis

adiwijaya@student.ub.ac.id

repository.ub.ac.id

ABSTRAK

Penggunaan aplikasi data teks pada saat ini sangat berkembang pesat, bahkan pada saat ini banyak aplikasi-aplikasi yang beralih pada *embedded system.* Hal ini dikarenakan fitur yang lebih efisien, murah, dan memiliki respon yang cepat sehingga tidak membahayakan kualitas data teks yang dikirimkan. Salah satu *embedded system* yang digunakan saat ini adalah penggunaan *personal digital assistant* atau PDA. Dibutuhkannya PDA sebagai alat komunikasi nirkabel menjadi hal yang sangat penting untuk diperhatikan, sebab data yang dikirimkan lewat komunikasi tersebut mungkin saja merupakan data perusahaan yang tidak dapat diketahui oleh siapapun. Penyadapan ponsel/PDA pada komunikasi dengan nirkabel kerap terjadi. Untuk mengatasi permasalahan yang terjadi, dibutuhkan suatu keamanan informasi untuk melengkapi kebutuhan keamanan data pada PDA/ponsel. Keamanan informasi yang digunakan untuk mengatasi permasalahan tersebut adalah menggunakan algoritma IDEA. Algoritma IDEA digunakan karena memberikan keamanan tingkat tinggi berdasarkan kunci rahasia yang kompleks. Langkah yang dilakukan untuk menerapkan algoritma ini adalah analisis kebutuhan sistem untuk mengetahui semua kebutuhan yang diperlukan oleh sistem yang akan dibangun dan diuji, setelah itu dilakukan perancangan sistem menggunakan FPGA Xilinx Spartan 3-e sebagai media simulasi sebelum dapat dirancang langsung pada *integrated circuit cryptoprocessor*, kemudian hasil perancangan sistem akan diimplementasikan pada FPGA dan akan ditampilkan pada PC. Setelah sistem diimplementasikan maka akan dilakukan pengujian dan analisis menggunakan pengujian fungsional, validitas, dan waktu eksekusi. Berdasarkan hasil yang didapatkan bahwa implementasi enkripsi data teks dengan metode IDEA dapat diimplementasikan pada FPGA Xilinx Spartan 3-e dan menghasilkan *ciphertext* untuk mengenkripsi data yang ada dan hasil pengujian menggunakan pengujian fungsional dan validitas memberikan hasil 100% sistem benar dan menghasilkan suatu enkripsi data teks dengan waktu enkripsi data teks sebesar 7.788ns.

Kata kunci: *embedded system, field programmable gate array (fpga), very high speed integrated circuit hardware description language (vhdl), international data encryption algorithm (idea).*

UNIVERSITAS BRAWIJAYA

repository.ub.ac.id

ABSTRACT

The use of text data applications is currently growing rapidly, even now many applications are switching to embedded systems. Many users choose to switch to embedded systems because they are more efficient, cheap, and have a fast response so it does not endanger the quality of the submitted text data. One of the embedded systems used today is the use of personal digital assistants or PDAs. The need for a PDA as a wireless communication tool is very important to be noticed, because the data sent through the communication may be company data that cannot be known by anyone. Tapping cellphones / PDAs on wireless communication often occurs. To solve the problem, information security is needed to complete the needs of the data security on PDAs / cellphones. Information security used to resolve these problems is using the IDEA algorithm. The IDEA algorithm is used because it provides high level security based on complex secret keys. The step taken to implement this algorithm is a system needs analysis to find out all the needs needed by the system to be built and tested, after that the system design is done using the Xilinx Spartan FPGA 3-e as a simulator before it can be designed directly on cryptoprocess integrated circuit, then the system design results will be implemented on the FPGA and will be displayed on the PC. After the system is implemented it will be tested and analyzed using functional testing, validity, and execution time. Based on the results obtained the text data encryption implementation with IDEA method can be implemented on FPGA Xilinx Spartan 3-e and generate ciphertext to encrypt existing data and test results using functional testing and validity give 100% correct system results and generate a text data encryption with text data encryption time of 7.788ns.

***Keyword** : embedded system, field programmable gate array (fpga), very high speed integrated circuit hardware description language (vhdl), international data encryption algorithm (idea).*

UNIVERSITAS BRAWIJAYA

DAFTAR ISI

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

1.1 Latar belakang

Hingga saat ini, komunikasi menjadi hal yang sangat penting bagi manusia sebagai makhluk sosial. Tidak hanya berkomunikasi secara langsung, komunikasi tidak langsung juga menjadi salah satu alternatif berkomunikasi seiring dengan berjalannya waktu seperti meningkatnya penggunaan aplikasi data teks *chatting* atau *e-mail*, bahkan untuk hal yang sifatnya penting. Menjadi perhatian khusus pada saat ini mengingat data teks yang penting, seperti data perbankan, data pelanggan dan lainnya dapat dikomunikasikan melalui aplikasi tersebut. Dalam hal ini, diperlukan suatu aplikasi yang menjamin keamanan pada data tersebut.

Menurut riset yang dilakukan oleh *Transparency Market Research*, aplikasi-aplikasi yang memerlukan data teks yang penting tersebut sekarang sudah ribuan jumlahnya, bahkan akhir-akhir ini banyak aplikasi-aplikasi yang beralih ke *embedded system*. Pengguna banyak yang memilih beralih ke *embedded system* karena lebih efisien, murah, dan memiliki respon yang cepat sehingga tidak membahayakan kualitas data teks yang dikirimkan. Salah satu penggunaan *embedded system* saat ini adalah penggunaan *personal digital assistant* atau PDA. PDA merupakan sebuah alat eletronik yang berbasis komputer dan berbentuk kecil yang dapat dibawa kemana-mana. Pada saat ini, PDA memiliki berbagai fitur seperti pengaksesan internet, penerimaan surat elektronik (*e-mail*), penerimaan radio dan pencatat memo. Fitur-fitur yang ada pada PDA sangat berhubungan dengan data yang harus dijaga keamanan dan kerahasiaan datanya. Namun, pada saat PDA melakukan pengiriman data melalui komunikasi dengan nirkabel, penyadapan/pencurian data kerap terjadi. Hal tersebut terjadi dikarenakan tidak terjaminnya keamanan yang menyebabkan ketika dilakukannya pengiriman data melalui komunikasi nirkabel yang tidak terenkripsi mengakibatkan data dapat dengan mudah disadap. Dalam hal ini diperlukan keamanan informasi untuk melengkapi kebutuhan keamanan data pada PDA/ponsel.

Kriptografi adalah suatu metode yang dapat digunakan untuk mengamankan informasi, karena kriptografi mengkodekan suatu informasi sedemikian rupa sehingga informasi tersebut tidak dapat diketahui oleh pihak-pihak yang tidak berhak mengetahuinya. Selain menjamin kerahasiaan informasi (*confidentiality*), teknik kriptografi juga dapat menjamin keutuhan data (*data integrity*), otentikasi (*authentication*) dan tahan terhadap penyangkalan (*non-repudation*) (Munir, Pengantar Kriptografi, 2004). Hingga saat ini kriptografi dapat diterapkan dengan mengimplementasikan suatu algoritma kriptografi baik dalam bentuk perangkat lunak maupun perangkat keras. Namun, Implementasi algoritma kriptografi pada *embedded system* yang berbasis perangkat keras yang dapat digunakan untuk mengamankan komunikasi data masih sedikit dijumpai. Sehingga, pada penelitian ini menitikberatkan pada implementasi algoritma kriptografi pada perangkat keras untuk pengamanan *embedded system* untuk menjamin

kerahasiaan informasi yang akan di komunikasikan. Salah satu metode kriptografi yang dianggap sebagai algoritma *block cipher* terbaik dan teraman yang tersedia untuk publik hingga saat ini yakni metode kriptografi IDEA (*International Data Encryption Algorithm*).

Algoritma IDEA atau *International Data Encryption Algorithm* adalah salah satu algoritma kriptografi yang mempunyai berbagai pengaplikasian dalam pengiriman data yang aman pada lingkup jaringan dan sistem pengukuran terdistribusi. Algoritma IDEA digunakan karena memberikan keamanan tingkat tinggi berdasarkan kunci rahasia yang kompleks, dapat diimplementasikan secara ekonomis dan efisien dalam komponen elektronik serta dilindungi paten untuk mencegah penipuan dan pembajakan algoritma tersebut (Chang, 2004). Berdasarkan penjelasan tersebut, dilakukannya implementasi algoritma IDEA untuk menjamin komunikasi data yang dilakukan pada *embedded system*. Sebelum dilakukan pengimplementasian langsung pada *integrated circuit cryptoprocessor*, terlebih dahulu dilakukan simulasi pada perangkat FPGA agar tidak terjadi kesalahan dalam pengimplementasian pada *integrated circuit* tersebut.

Field Programmable Gate Array (FPGA) merupakan sebuah *integrated circuit digital* yang sering digunakan untuk mengimplementasikan rangkaian digital. FPGA digunakan untuk meningkatkan efisiensi rancangan dengan cara mengurangi pemakaian pemrograman perangkat lunak (*software*). FPGA mempunyai koreksi *error* yang kecil dan merupakan teknologi yang bebas (*technology-independent*) untuk diimplementasikan dalam berbagai algoritma. Dalam hal ini, FPGA dipilih sebagai media simulasi yang nantinya hasil dari implementasi pada FPGA dapat menjadi acuan ketika akan diimplementasikan pada *integrated circuit cryptoprocessor*. Dengan dilakukan hal tersebut kesalahan ketika mendesain *integrated circuit cryptoprocessor* menjadi sangat minim karena sudah disimulasikan sebelumnya. Sehingga implementasi *integrated circuit cryptoprocessor* dapat menyelesaikan permasalahan pada komunikasi *embedded system*. Oleh karena itu, judul penelitian ini adalah "Perancangan dan Implementasi Algoritma Enkripsi IDEA (*International Data Encryption Algorithm*) pada Perangkat Kriptografi Berbasis FPGA (*Field Programmable Gate Array*)."

1.2 Rumusan masalah

Berdasarkan permasalahan yang telah di paparkan pada latar belakang, maka rumusan masalah pada penelitian ini sebagai berikut:

1. Bagaimana perancangan enkripsi data text dengan algoritma *International Data Encryption Algorithm* pada perangkat kriptografi berbasis FPGA?
2. Bagaimana pengimplementasian enkripsi data text dengan algoritma *International Data Encryption Algorithm* pada perangkat kriptografi berbasis FPGA?
3. Bagaimana pengujian dan analisis fungsional sistem, validitas dan waktu enkripsi data text dengan algoritma *International Data Encryption Algorithm* pada perangkat kriptografi berbasis FPGA?

1.3 Tujuan

Tujuan dari penelitian ini antara lain:

1. Merancang enkripsi data text dengan algoritma *International Data Encryption Algorithm* pada perangkat kriptografi berbasis FPGA.
2. Mengimplementasikan enkripsi data text dengan algoritma *International Data Encryption Algorithm* pada perangkat kriptografi berbasis FPGA.
3. Menguji dan menganalisis fungsional sistem, validitas dan waktu enkripsi data text dengan algoritma *International Data Encryption Algorithm* pada perangkat kriptografi berbasis FPGA.

1.4 Manfaat

Penulisan skripsi ini diharapkan mempunyai manfaat yang baik dan berguna bagi pembaca dan penulis. Adapun manfaat yang diharapkan adalah sebagai berikut:

A. Bagi Penulis

   1. Sebagai media untuk pengimplementasian ilmu pengetahuan teknologi yang telah didapat selama mengikuti perkuliahan di Teknik Komputer Universitas Brawijaya.
   2. Penulis mendapatkan pengetahuan dan wawasan terkait kriptografi dan *embedded system* yang terdapat pada penelitian ini dengan menggunakan FPGA.

B. Bagi Pembaca

   1. Pembaca mendapatkan wawasan dan pengimplementasian enkripsi data text dengan algoritma *International Data Encryption Algorithm* pada perangkat kriptografi berbasis FPGA.
   2. Sebagai bahan dan acuan dalam mengembangkan dan melanjutkan riset yang memiliki keterkaitan dengan penelitian ini.

1.5 Batasan masalah

Berdasarkan rumusan masalah tersebut dikhususkan lagi sesuai dengan beberapa batasan – batasan seperti berikut:

1. Implementasi pada perangkat keras menggunakan *Field Programmable Gate Array* Xilinx Spartan-3e Development Boards dengan menggunakan bahasa pemrograman *Very High Speed Integrated Circuit Hardware Description Language*.
2. Penelitian ini hanya dititik beratkan pada proses Enkripsi
3. Data yang digunakan adalah data teks dengan panjang 64-bit untuk *plaintext* dan 64-bit untuk *ciphertext* dalam bentuk HEX.

1.6 Sistematika pembahasan

Bagian ini berisi struktur skripsi ini mulai Bab Pendahuluan sampai Bab Penutup dan deskripsi singkat dari masing-masing bab. Diharapkan bagian ini dapat membantu pembaca dalam memahami sistematika pembahasan isi dalam skripsi ini.

BAB I PENDAHULUAN

Bab I Pendahuluan menguraikan latar belakang dari skripsi, rumusan masalah tentang pengembangan penelitian, tujuan dan manfaat skripsi serta sistematika penyusunan laporan skripsi.

BAB II LANDASAN KEPUSTAKAAN

Bab II menguraikan sub-bab yang berisi tinjauan pustaka sebagai dasar penelitian. serta dasar teori yang berisi teori-teori pendukung penelitian, antara lain algoritma *International Data Encryption Algorithm* (IDEA), *Field Programmable Gate Array* (FPGA) dan *VHSIC Hardware Description Language* (VHDL).

BAB III METODE PENELITIAN

Bab III menjelaskan tentang metodologi penelitian yang akan digunakan dalam penelitian ini yang berisi studi literatur, dan langkah-langkah apa yang dilakukan dalam penelitian. Langkah-langkah yang dilakukan dalam penelitian ini berisi studi dan pengkajian literatur, melakukan analisis kebutuhan sistem, melakukan perancangan sistem, implementasi sistem, pengujian dan analisis, dan menarik kesimpulan dan saran dari penelitian yang dilakukan.

BAB IV IMPLEMENTASI SISTEM

Bab V menguraikan proses implementasi dari dasar teori yang telah dipelajari sesuai analisis dan perancangan sistem yang telah dibuat, yang memuat lingkungan implementasi dari enkripsi teks dengan algoritma IDEA, batasan implementasi yang dilakukan, implementasi perangkat lunak dan implementasi perangkat keras.

BAB V PENGUJIAN DAN ANALISIS SISTEM

Bab VI memuat hasil pengujian dan analisis terhadap sistem yang telah direalisasikan untuk meningkatkan keamanan informasi pada perangkat kriptografi berbasis FPGA, yang memuat skenario pengujian, hasil pengujian menggunakan pengujian fungsional, validitas dan waktu eksekusi.

BAB VI PENUTUP

Bab VII memuat kesimpulan yang diperoleh dari pembahasan keseluruhan bab dan saran hasil analisis yang dapat digunakan untuk pengembangan penelitian yang lebih lanjut.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Tinjauan pustaka

Pada penelitian yang dilakukan, penulis melakukan kajian pustaka dari beberapa penelitian sebelumnya mengenai topik perancangan dan implementasi dari perangkat kriptografi yang digunakan sebagai referensi untuk menyelesaikan penelitian ini. Literatur yang didapatkan oleh penulis berupa artikel yang berisi teori-teori dan penelitian-penelitian yang dilakukan oleh peneliti sebelumnya. Pada artikel dan penelitian sebelumnya membantu penulis dalam memahami bagaimana konsep dasar *embedded system*, arsitektur pada FPGA menggunakan metode yang telah ada.

Pada tinjauan pustaka ini membahas empat penelitian yang sudah dilakukan sebelumnya. Yang pertama adalah milik Ms Snehal Patil dan Prof. Vrunda Bhusari dari BSIOTR Pune, Departement of Computer yang berjudul "An Enhancement In International Data Encryption Algorithm For Increasing Security", pada tahun 2014. Pada penelitian ini membahas mengenai pengembangan dari algoritma *The Data Encryption Standard* (DES) yang mana telah populer pada tahun 1991 dengan kunci rahasia dari enkripsi algoritma tersebut dan telah digunakan oleh banyak perusahaan komersil dan pengaplikasian pada finansial produk. Pada tahun 1976, algoritma tersebut telah teruji aman dari segala macam bentuk ancaman dari analis kriptografi. Namun, ukuran kunci yang menjadi standar dari algoritma tersebut terbilang kecil dan dapat dipecahkan dalam waktu 22 jam. Dari penulisan tersebut, diperkenalkan algoritma *International Data Encryption Algorithm* sebagai pengganti dari algoritma DES dilihat dari keamanan desain algoritma dan memiliki reputasi yang baik.

Penelitian yang kedua adalah penelitian yang di miliki oleh Rajashekhar Modugu, Yong-Bin Kim, dan Minsu Choi, mahasiswa dari Universitas Missouri, Departemen Teknik Elektro dan Komputer berjudul "Design and Performance Measurement of Efficient IDEA (International Data Encryption Algorithm) Crypto-Hardware using Novel Modular Arithmetic Components". Penelitian ini dilakukan pada tahun 2012 dengan fokus terhadap efisiensi algoritma IDEA yang di aplikasikan dalam perangkat kriptografi berbasis perangkat keras. Pada penelitian tersebut menjelaskan bahwa algoritma IDEA dapat diimplementasikan pada perangkat keras dengan beberapa penjelasan terkait dengan analisis penilaian seperti efisiensi pada algoritma tersebut.

Penelitian yang ketiga adalah penelitian milik Andreas Dandalis dan Viktor K.Prasanna dengan judul penelitiannya "Configuration Compression for FPGA-Based Embedded System". Penelitian ini dilakukan pada tahun 2005. Pada penelitian ini pula mengusulkan bahwa FPGA merupakan teknologi yang menjanjikan untuk mengembangkan *embedded system* berkinerja tinggi, sehingga dapat melakukan komputasi yang kompleks pada algoritma IDEA.

Penelitian yang keempat adalah penelitian milik Allen Michalski, Kris Gaj, dan Tarek El-Ghazawi dari Universitas George Mason dengan judul penelitianya "An

Implementation Comparison of an IDEA Encryption Cryptosystem on Two General-Purpose Reconfigurable Computers" pada tahun 2003. Pada penelitian ini membahas perbandingan implementasi enkripsi algoritma IDEA pada FPGA dan perangkat komputer SRC-6E. Dalam penelitian itu, berdasarkan perancangan desain algoritma IDEA yang telah dibuat oleh peneliti tersebut menggunakan total *Slice* sebanyak 17.292 (51%), total *LUT* sebanyak 7639 (11%), total *Multi18x18* sebanyak 136 (94%) dan memiliki waktu uji algoritma IDEA sebesar 12.191ns.

2.2 Dasar teori

2.2.1 Field Programmable Gate Array

Field-Programmable Gate Array (FPGA) merupakan sebuah IC digital yang sering digunakan untuk mengimplementasikan rangkaian digital. FPGA berbentuk komponen elektronika dan semikonduktor yang terdiri dari komponen gerbang terprogram (*programmable logic*) dan juga sambungan terprogram (interkoneksi). Komponen gerbang terprogram yang dimiliki meliputi jenis gerbang logika biasa (*AND, OR, NOT*) maupun jenis fungsi matematis dan kombinatorik yang lebih kompleks, seperti *decoder, adder, subtractor, multiplier*, dll (Ian Kuon, 2008). Blok-blok komponen di dalam FPGA bisa juga mengandung elemen memori (*register*) mulai dari *flip-flop* sampai pada RAM (*Random Access Memory*). FPGA sangat sesuai untuk pemrosesan komputasi dari algoritma integrasi numerik. Keuntungan implementasi FPGA digunakan untuk meningkatkan efisiensi rancangan dengan cara mengurangi pemakaian pemrograman perangkat lunak (*software*). FPGA mempunyai koreksi error yang kecil dan merupakan teknologi yang bebas (*technology-independent*) untuk diimplementasikan dalam berbagai algoritma.

Kinerja aplikasi FPGA lebih cepat dibandingkan dengan aplikasi mikrokontroler, karena FPGA hanya mensintesis perangkat keras (*hardware*) saja, sementara mikrokontroler mengeksekusi instruksi perangkat lunak (*software*) yang digunakan untuk mengendalikan perangkat keras (*hardware*), sehingga waktu tunda yang diimplementasikan hanya memakan waktu tunda perambatan (*propagation delay*) saja (Prasanna, 2005). Pemodelan FPGA membutuhkan informasi terkait dengan tingkat perbedaan abstraksi dan jenis model yang digunakan. Seorang perancang FPGA harus mampu mengambil beberapa tahapan pemodelan untuk memastikan hasil model rancangannya melalui model simulasi yang telah disediakan oleh vendor FPGA masing-masing.

Pengertian terprogram (*programmable*) dalam FPGA adalah mirip dengan interkoneksi saklar dalam breadboard yang bisa diubah oleh pembuat desain sesuai kebutuhan pengguna. Dalam FPGA, interkoneksi ini bisa diprogram kembali oleh pengguna maupun pendesain di dalam lab atau lapangan (*field*). Oleh karena itu jajaran gerbang logika (*Gate Array*) ini disebut *field-programmable*. Jenis gerbang logika yang bisa diprogram meliputi semua gerbang dasar untuk memenuhi kebutuhan yang manapun.

Vendor-vendor FPGA berbasis *static random access memory* (SRAM) dibuat oleh Xilinx Inc., Altera Corp., Atmel dan Lattice Semiconductor; sedangkan, vendor-vendor FPGA berbasis *flash* dan *antifuse* dibuat oleh Actel Corp. dan Quick Logic Corp. Pemain lainnya yang kemudian pupus di tengah jalan diantaranya adalah Intel, Texas Instrument, Motorola, NSC, AMD, Cypress, Philips. Pendatang dalam dunia FPGA yang telah diserap dan gagal dalam pemasaran produknya adalah Dynachip, PlusLogic, Triscend, SiliconSpice, Chameleon, Quicksilver, Morphics, Adaptive Silicon. Kecepatan inovasi dalam dunia FPGA ditentukan oleh vendor yang memimpin pemasaran produknya. Dua vendor FPGA yang sering dipakai oleh perancang adalah Xilinx, Inc. dan Altera Corp.

2.2.1.1 Arsitektur FPGA

Sebuah IC FPGA terdiri dari tiga komponen pendukung utamanya (Xilinx, 2013) yakni:

1. *Configurable Logic Block* (CLB)
   CLB merupakan komponen dasar yang membentuk IC FPGA berupa matrik-matrik yang saling terhubung oleh PI, yang dapat dikonfigurasikan untuk melakukan rangkaian logika kombinasional, *shift register*, atau *Random Access Memory*.
2. *Input Output Block* (IOB)
   IOB merupakan penghubung atau sebagai antarmuka antara pin-pin terminal IC FPGA dengan kawat penghubung atau jalur-jalur koneksi diluar IC, IOB dikelompokkan ke dalam beberapa I/O Bank yang sesuai dengan standar I/O.
3. *Programmable Interconnect* (PI)
   PI merupakan komponen yang berperan sebagai kawat atau saklar penghubung yang dapat dikonfigurasikan dan mengelilingi blok-blok CLB, yang akan menghubungkan antar blok-blok CLB maupun dengan IOB.

Selain CLB, IOB, PI, beberapa IC FPGA juga dilengkapi dengan elemen-elemen lain seperti RAM dan *Delay-Locked Loop* (DLL).

2.2.1.2 Mengkonfigurasikan FPGA

Tanpa memperhatikan metode interkoneksi yang digunakan pada FPGA, dapat dilihat jelas bahwa untuk membayangkan saklar mana yang harus dibuka dan yang ditutup untuk membuat suatu rangkaian logika merupakan pekerjaan yang sangat berat, karenanya perusahaan pembuat chip menyediakan perangkat lunak yang melakukan deskripsi dari desain logika sebagai masukkannya dan kemudian akan menghasilkan file biner yang akan mengkonfigurasikan saklar-saklar didalam chip CPLD atau FPGA sehingga menjadi seperti desain yang dibuat.

Perusahaan pembuat chip pada umumnya memberikan perangkat lunak secara cuma-cuma atau gratis. Perangkat lunak ini digunakan untuk mendukung proses *design entry*, *simulation*, *synthesis, place-and-route*, dan *programming through special cable* (JTAG).

Berdasarkan buku *Spartan-3e FPGA Family Data Sheet,* Implementasi sebuah desain logika dengan perangkat lunak tersebut biasanya terdiri dari beberapa langkah :

1. Mendeskripsikan rangkaian logika dari sistem yang dirancang dengan menggunakan *hardware description language* (HDL) seperti VHDL atau *verilog* atau dengan menggambarkannya dengan *schematic editor*.
2. Deskripsi dari sistem dikonversikan menjadi sebuah *Netlist* menggunakan *program logic synthesizer*.
3. Implementasi rangkaian dengan menggunakan pemetaan gerbang logika dan intekoneksi (*routing*) *netlist* kedalam FPGA menggunakan *implementation tools*.
4. Setelah fase implementasi selesai, aplikasi akan membuat sebuah file *bitstream* yang berisi nilai logika '1' dan '0' sebagai representasi konfigurasi rangkaian digital yang akan diimplementasikan ke dalam FPGA untuk menunjukkan terbuka atau tertutupnya saklar pada IC FPGA.
5. File *bitstream* kemudian diunggah ke dalam IC FPGA atau IC memori yang tersedia melalui kabel JTAG sesuai dengan jenis dan tipe *development board* yang digunakan.

2.2.1.3 Spesifikasi FPGA Xilinx Spartan 3-e

Spesifikasi FPGA Xilinx Spartan 3-e yang digunakan pada penulisan skripsi ini adalah sebagai berikut:

1. Mendukung FPGA keluarga Spartan-3e & CoolRunner-II
2. Xilinx *platform flash*.
3. 50 MHz *crystal clock oscillator*.
4. 128 Mbit parallel *flash*, 16 Mbit SPI *flash*, 64 Mbyte DDR SDRAM.
5. Ethernet 10/ 100 Phy.
6. JTAG USB *download*.
7. 2 port serial RS-232 dan 1 port PS/ 2 mouse/ keyboard.
8. Rotary encoder with push button, 4 slides switch, 8 LED output, 4 momentary -contact push button.
9. 100-pin hirose *expansion connection port* dan 3 buah 6-pin *expansion connector*.
10. LCD 16 *character* x 2 line.

2.2.2 Kriptografi

Kriptografi berasal dari bahasa Yunani yakni *kriptos* yang artinya tersembunyi dan *graphia* yang artinya sesuatu yang tertulis, sehingga kriptografi dapat disebut sebagai sesuatu yang tertulis secara rahasia. Menurut (Munir, Pengantar Kriptografi, 2004), Kriptografi merupakan suatu bidang ilmu yang mempelajari tentang bagaimana merahasiakan suatu informasi penting ke dalam suatu bentuk yang tidak dapat dibaca oleh siapapun serta mengembalikannya kembali menjadi informasi semula dengan menggunakan berbagai macam teknik yang telah ada sehingga informasi tersebut tidak dapat diketahui oleh pihak manapun

yang bukan pemilik atau yang tidak berkepentingan. Sisi lain dari kriptografi ialah kriptanalisis (*Cryptanalysis*) yang merupakan studi tentang bagaimana memecahkan mekanisme kriptografi. Bagi kebanyakan orang, kriptografi lebih diutamakan dalam menjaga komunikasi tetap rahasia dan khusus. Seperti yang telah diketahui dan disetujui bahwa perlindungan (proteksi) terhadap komunikasi yang sensitif telah menjadi penekanan kriptografi selama ini. Akan tetapi hal tersebut hanyalah sebagian dari penerapan kriptografi dewasa ini.

2.2.2.1 Terminologi

Beberapa terminologi atau istilah yang penting untuk diketahui didalam kriptografi menurut (Munir, Pengantar Kriptografi, 2004) dalam bukunya yang berjudul Kriptografi adalah sebagai berikut.

A. Pesan, Plainteks, dan Cipherteks
   Pesan (message) adalah data atau informasi yang dapat dibaca dan dimengerti maknanya. Nama lain untuk pesan adalah plainteks (*plaintext*) atau teks-jelas (*cleartext*). Pesan tidak hanya berupa teks, tetapi juga dapat berbentuk gambar (*image*), suara (*audio*), *video*, atau berkas biner lainnya. Pesan perlu disandikan ke bentuk lain yang tidak dipahami agar tidak dapat dimengerti maknanya oleh pihak lain. Bentuk pesan yang tersandi disebut cipherteks (*ciphertext*) atau kriptogram (*cryptogram*). Cipherteks harus dapat ditransformasikan kembali menjadi plainteks semula agar pesan yang diterima bisa dibaca.
B. Pengirim dan Penerima
   Komunikasi data melibatkan pertukaran pesan antara dua entitas. Pengirim (*sender*) adalah entitas yang mengirim pesan kepada entitas lainnya. Penerima (*receiver*) adalah entitas yang menerima pesan. Entitas di sini dapat berupa orang, komputer, kartu kredit, dan sebagainya.
C. Enkripsi dan Dekripsi
   Proses menyandikan plainteks menjadi cipherteks disebut enkripsi (*encryption*) atau *enciphering* (standar nama menurut ISO 7498-2). Sedangkan proses mengembalikan cipherteks menjadi plainteks semula disebut dekripsi (*decryption*) atau *deciphering* (standard nama menurut ISO 7498-2). Enkripsi adalah proses mengamankan suatu informasi dengan membuat informasi tersebut tidak dapat dibaca tanpa bantuan pengetahuan khusus. Keuntungan dari enkripsi adalah kode asli kita tidak dapat dibaca oleh orang lain. Dekripsi adalah proses mengembalikan suatu informasi dengan cara tertentu dan sesuai dengan algoritma enkripsi yang dipakai. Dekripsi merupakan proses kebalikan dari proses enkripsi, mengubah *ciphertext* kembali ke dalam bentuk *plaintext*. Proses utama dalam suatu algoritma kriptografi adalah enkripsi dan deskripsi.
D. Cipher dan kunci
   Cipher atau algoritma kriptografi adalah aturan untuk *enchipering* dan *dechipering*, atau fungsi matematika yang digunakan untuk enkripsi dan dekripsi. Beberapa cipher memerlukan algoritma yang berbeda untuk *enchipering* dan *dechipering*. Kunci (*key*) adalah parameter yang digunakan

untuk tranformasi *enchipering* dan *dechipering*. Kunci biasanya berupa string atau deretan bilangan.

E. Penyadap
Penyadap (*eavesdropper*) adalah orang yang mencoba menangkap pesan selama ditransmisikan. Tujuan penyadap adalah untuk mendapatkan informasi sebanyak-banyaknya mengenai sistem kriptografi yang digunakan untuk berkomunikasi dengan maksud untuk memecahkan cipherteks.

F. Kriptanalisis dan Kriptologi
Kriptanalisis (*cryptanalysis*) adalah ilmu dan seni untuk memecahkan cipherteks menjadi plainteks tanpa mengetahui kunci yang digunakan. Pelakunya disebut kriptanalis. Kriptologi (*cryptology*) adalah studi mengenai kriptografi dan kriptanalisis.

2.2.2.2 Tujuan Kriptografi

Tujuan dari kriptografi yang juga merupakan aspek keamanan informasi adalah sebagai berikut (Munir, Pengantar Kriptografi, 2004).

A. Kerahasiaan (*confidentiality*)
Kerahasiaan bertujuan untuk melindungi suatu informasi dari semua pihak yang tidak berhak atas informasi tersebut. Terdapat beberapa cara yang dapat digunakan untuk menjaga kerahasiaan suatu informasi, mulai dari penjagaan secara fisik misalnya menyimpan data pada suatu tempat khusus sampai dengan penggunaan algoritma matematika untuk mengubah bentuk informasi menjadi tidak terbaca.

B. Integritas data (*data integrity*)
Integritas data bertujuan untuk mencegah terjadinya pengubahan informasi oleh pihak-pihak yang tidak berhak atas informasi tersebut. Untuk menjamin integritas data ini kita harus mempunyai kemampuan untuk mendeteksi terjadinya manipulasi data oleh pihak-pihak yang tidak berkepentingan. Manipulasi data yang dimaksud di sini meliputi penyisipan, penghapusan, maupun penggantian data.

C. Otentikasi (*authentication*)
Otentikasi merupakan identifikasi yang dilakukan oleh masing – masing pihak yang saling berkomunikasi, maksudnya beberapa pihak yang berkomunikasi harus mengidentifikasi satu sama lainnya. Informasi yang didapat oleh suatu pihak dari pihak lain harus diidentifikasi untuk memastikan keaslian dari informasi yang diterima. Identifikasi terhadap suatu informasi dapat berupa tanggal pembuatan informasi, isi informasi, waktu kirim dan hal-hal lainnya yang berhubungan dengan informasi tersebut.

D. Non-repudiation
Non-repudiation berfungsi untuk mencegah terjadinya penyangkalan terhadap suatu aksi yang telah dilakukan oleh pelaku aksi itu sendiri. Jika terjadi penyangkalan maka diperlukan suatu prosedur yang melibatkan pihak ketiga untuk menyelesaikan masalah tersebut.

2.2.2.3 Jenis-jenis Kriptografi

Berdasarkan kunci yang dipakai untuk enkripsi dan dekripsi, kriptografi dapat dibedakan atas dua golongan, yaitu :

A. Kriptografi Kunci Simetris (*symmantric key cryptography*)

B. Kriptografi Kunci Asimetris (*asymmetric key cryptography*)

2.2.2.4 Kriptografi Kunci Simetris

Algoritma kriptografi kunci simeteris atau disebut juga algoritma kriptografi konvensioanal adalah algoritma yang menggunakan kunci untuk proses enkripsi sama dengan kunci yang digunakan untuk melakukan proses dekripsi (Suprapti, 2003). Algoritma kriptografi simeteris dibagi menajdi 2 kategori yaitu algoritma aliran (*Stream Ciphers*) dan algoritma blok (*Block Ciphers*). Pada algoritma aliran, proses penyandiannya berorientasi pada satu bit atau satu byte data. Sedang pada algoritma blok, proses penyandiannya berorientasi pada sekumpulan bit atau byte data (per blok). Contoh algoritma kunci simetris yang terkenal adalah DES (*Data Encryption Standard*), IDEA (*International Data Encryption Algorithm*). Metode yang paling umum untuk kriptografi kunci rahasia adalah *block ciphers, stream ciphers, desain cipher*.

A. Block Cipher

Block cipher adalah bentuk algoritma enkripsi kunci simetri yang mentransformasikan satu blok data tertentu dari *plaintext* ke dalam satu blok data *ciphertext* dengan panjang blok yang sama. Transformasi ini berlangsung melalui penggunaan kunci rahasia yang disediakan oleh pemakai (*user*). Dekripsi dilakukan dengan menggunakan transformasi kebalikan terhadap blok *ciphertext* menjadi satu blok *plaintext* dengan kunci dan panjang blok yang sama. Panjang blok tertentu disebut ukuran blok (*block size*) dimana ukuran blok tersebut bervariasi misalnya 16 bit, 32 bit, 64 bit, 128 bit atau 256 bit tergantung dari teknik yang digunakan dan perkembangan kemampuan mikroprosesor selanjutnya.

Karena blok *plaintext* yang berbeda dipetakan ke blok *ciphertext* yang berbeda (untuk memungkinkan dekripsi yang unik), suatu *block cipher* secara efektif menyediakan satu permutasi (korespondensi satu ke banyak) dari set pesan yang mungkin. Permutasi berpengaruh pada saat enkripsi tertentu yang sudah pasti rahasia, karena permutasi tersebut adalah fungsi dari kunci rahasia. Jika kita menggunakan satu *block cipher* untuk mengenkrip satu pesan dengan panjang sembarang, kita menggunakan teknik yang dikenal sebagai modus operasi untuk *block cipher* tersebut. Agar dapat berguna, satu modus operasi setidak-tidaknya seefisien dan seaman *cipher fundamental.* Teknik enkripsi mungkin memiliki sifat-sifat tambahan terhadap sifat-sifat dasar yang dimiliki teknik biasa. Teknik standar DES telah dipublikasi dalam berbagai publikasi. Versi standar yang lebih umum menggabungkan 4 modus operasi dari DES untuk dapat diaplikasikan terhadap *block cipher* dengan ukuran blok sembarang.

B. *Stream Cipher*

Merupakan jenis algoritma enkripsi simetri yang mentransformasikan data secara karakter per karakter. *Stream ciphers* dapat dibuat sangat cepat sekali, jauh lebih cepat dibandingkan dengan algoritma *block cipher* yang manapun. Sementara algoritma *block cipher* secara umum digunakan untuk unit *plaintext* yang berukuran besar sedangkan *stream cipher* digunakan untuk blok data yang lebih kecil, biasanya ukuran bit. Proses enkripsi terhadap *plaintext* tertentu dengan algoritma *block cipher* tentu akan menghasilkan *ciphertext* yang sama jika kunci yang sama digunakan. Dengan *stream cipher*, transformasi dari unit *plaintext* yang lebih kecil ini berbeda antara satu dengan lainnya, tergantung pada kapan unit tersebut ditemukan selama proses enkripsi.

Suatu *stream cipher* akan menghasilkan apa yang disebut suatu *keystream* yaitu suatu barisan bit yang digunakan sebagai kunci. Proses enkripsi dicapai dengan menggabungkan *keystream* dengan *plaintext* biasanya dengan operasi *bitwise* XOR

C. *Desain Cipher*

Terdapat dua prinsip dasar untuk menghasilkan cipher yang aman, yaitu *confusion* dan *diffusion*. Tanpa memperhatikan hal ini, cipher kita mungkin akan sangat mudah dipecahkan sandinya.

Confusion berarti mengaburkan hubungan antara *plaintext* dan *ciphertext*. Ini akan membuat frustasi usaha untuk mencari keteraturan dan pola statistik antara *plaintext* dan *ciphertext*. Cara paling mudah untuk melakukan hal ini adalah dengan substitusi. Substitusi modern menggunakan cara yang sangat komplek. Namun cara ini belum cukup. Cipher Jerman, Enigma, yang menggunakan algoritma substitusi yang komplek dipecahkan oleh Sekutu dalam perang dunia kedua.

Dimana *diffusion* berarti menghilangkan redundansi *plaintext* dengan menyebarkan masukan ke seluruh *ciphertext*. Diperlukan waktu yang lebih lama untuk memecahkan sandi rahasia ini, bila *diffusion* digunakan. Cara paling mudah untuk melakukan *diffusion* adalah transposisi atau permutasi. Dalam dunia kriptografi modern, *confusion* dan *diffusion* ini dilakukan secara sangat intensif dengan bantuan komputer.

2.2.2.5 Kriptografi Kunci Asimetri

Algoritma kriptografi asimetri adalah algoritma yang menggunakan kunci yang berbeda untuk proses enkripsi dan dekripsinya. Algoritma ini disebut juga algoritma kunci umum (*public key algorithm*) karena kunci untuk enkripsi dibuat umum (*public key*) atau dapat diketahui oleh setiap orang, tapi kunci untuk dekripsi hanya diketahui oleh orang yang berwenang mengetahui data yang disandikan atau disebut kunci pribadi (*private key*) (Suprapti, 2003).

Kriptografi kunci publik diperkenalkan oleh Whitfield Diffie dan Martin Hellman pada tahun 1976. Kriptografi kunci publik memiliki dua penggunaan utama, yakni enkripsi dan tanda tangan digital (*encryption and digital signatures*). Dalam sistem kriptografi kunci publik, masing-masing pihak

mendapat sepasang kunci, satu disebut kunci publik (*public key*) dan satu lagi disebut kunci rahasia (*private key*). Kunci publik dipublikasikan, sementara kunci rahasia tetap dirahasiakan. Keharusan penggunaan kunci secara bersama antara pengirim dan penerima pesan rahasia dihilangkan, semua komunikasi hanya melibatkan kunci publik, dan tidak ada kunci rahasia yang ditransmisikan atau digunakan bersama. Dalam sistem ini, tidak ada lagi kecurigaan terhadap keamanan dari sistem komunikasi. Satu-satunya kebutuhan bahwa kunci publik dikaitkan dengan penggunanya dalam lingkup yang saling mempercayai (contoh dalam suatu *trusted directory*). Seseorang dapat mengirimkan pesan rahasia dengan hanya menggunakan informasi yang umum (kunci publik), tetapi pesan tersebut hanya mungkin didekrip dengan menggunakan kunci rahasia, dimana satu-satunya yang memiliki kunci rahasia tersebut hanyalah orang yang diharapkan menerima pesan tersebut. Kriptografi kunci publik tidak hanya digunakan untuk merahasiakan pesan, tetapi juga untuk otentikasi (tanda tangan digital) dan teknik lainnya.

Dalam kriptografi kunci publik, kunci rahasia selalu berhubungan secara matematis terhadap kunci publik. Oleh karena itu, selalu dimungkinkan untuk menembus (menyerang) sistem kunci publik dengan menurunkan kunci rahasia dari kunci publik. Biasanya, cara untuk menangkal kemungkinan tersebut adalah membuat sesulit mungkin untuk menghasilkan kunci privat dari kunci publik. Sebagai contoh, beberapa kriptosistem kunci publik dibuat sedemikian hingga penurunan kunci rahasia (*private*) dari kunci publik mengharuskan penyerang melakukan faktorisasi terhadap bilangan yang sangat besar, dalam hal ini sangat sulit untuk melakukan penurunan. Inilah ide di belakang RSA *public-key cryptosystem*. Contoh algoritma terkenal yang menggunakan kunci asimetris adalah RSA dan ECC.

2.2.2.6 Kriptoprosesor

Kriptoproseor adalah sebuah chip yang di dalamnya terdapat atau berisi operasi kriptografi. Selain itu, kriptoprosesor juga mempunyai sistem yang tertanam dalam kemasan dengan beberapa langkah pengamanan fisik, sehingga hal ini memberikan tingkat resistensi yang baik. Tujuan dari kriptoprosesor sendiri adalah untuk bertindak sebagai gerbang keamanan dari subsistem keamanan, menghilangkan kebutuhan untuk melindungi sisa subsistem dengan tindakan keamanan fisik. Kriptografi banyak digunakan pada kartu pintar (smart card) seperti kartu operator telepon selular. Kriptografi menerima instruksi masukan dalam bentuk terenkripsi. setelah itu, kriptoprosesor melakukan dekripsi sehingga masukan tersebut dapat dikenali oleh program yang menjalankannya.

2.2.3 Algoritma IDEA

Algoritma IDEA atau *International Data Encryption Algorithm* adalah salah satu algoritma kriptografi yang mempunyai berbagai pengaplikasian dalam pengiriman data yang aman pada lingkup jaringan dan sistem pengukuran terdistribusi (Massey, 1993). Algoritma IDEA berawal dan berkembang dari

beberapa algoritma sebelumnya, yakni algoritma *Proposed Encryption Standard* dan algoritma *Improved Proposed Encryption Standard* pada tahun 1992. Algoritma IDEA sebagai algoritma simetris dengan operasi blok *plaintext* 64-bit dan menggunakan kunci 128-bit terdapat pula 8-round dan beroperasi pada sub blok 16-bit dengan menggunakan perhitungan aljabar sehingga algoritma IDEA dapat digunakan untuk implementasi pada perangkat kriptografi berbasis FPGA. Operasi yang digunakan tersebut terdapat pula penjumlahan modulo 2^{16}, perkalian modulo (2^{16} + 1), dan logika XOR (Osama Almasri, 2013).

2.2.3.1 Deskripsi umum algoritma IDEA

Algoritma utama dari sistem kriptografi IDEA adalah sebagai berikut (Chang, International Data Encryption Algorithm, 2004):

1. Proses enkripsi : $e_k(M) = C$
2. Proses dekripsi : $d_k(C) = M$

Dimana :

e = adalah fungsi enkripsi

d = adalah fungsi dekripsi

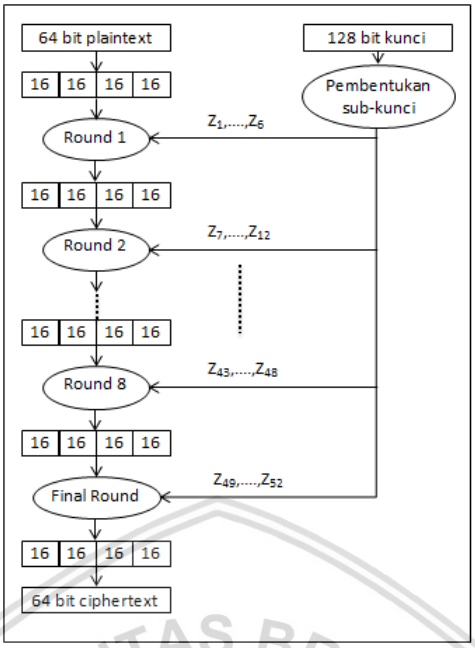
M = adalah *plaintext*

C = adalah *ciphertext*

K = adalah kunci enkripsi atau dekripsi

IDEA (*International Data Encryption Algorithm*) merupakan algoritma simetris yang beroperasi pada sebuah blok pesan *plaintext* terbuka dengan panjang blok 64-bit. Memilih panjang blok sebanyak 64 bit karena dirasa sudah cukup panjang bila kita akan melakukan analisis statistikal. Menggunakan kunci yang sama berukuran 128-bit, untuk proses enkripsi maupun dekripsi. Kunci yang digunakan panjangnya 128 bit karena bila dilakukan serangan *brute force* pada IDEA menggunakan sebuah sistem yang dapat melakukan pengujian 109 kunci per detiknya, akan diperlukan waktu 1013 tahun untuk mencari kuncinya. Pesan rahasia yang dihasilkan oleh algoritma ini (*chipertext*) berupa blok pesan rahasia dengan lebar satu blok 64-bit. Sebuah sistem yang dapat melakukan pengujian 109 kunci akan mencari semua kemungkinan dari 256 kunci DES dalam waktu di bawah satu detik (72 millidetik).

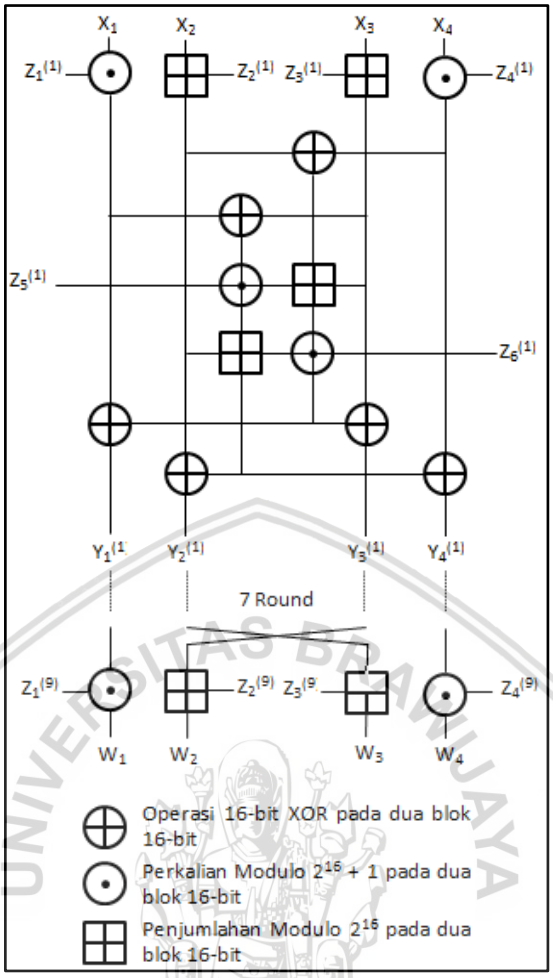
Dalam melakukan enkripsi menggunakan delapan putaran. Setiap putarannya digunakan enam kunci. Struktur dan skema dari IDEA adalah :



Gambar 2. 1 Struktur dan Skema Algoritma IDEA

Pesan Dekripsi menggunakan blok penyandi yang sama dengan blok proses enkripsi dimana kunci dekripsinya diturunkan dari kunci enkripsi. Algoritma ini menggunakan operasi campuran dari tiga operasi aljabar yang berbeda, yaitu XOR, operasi penjumlahan modulo 2^{16} dan operasi perkalian modulo $(2^{16} + 1)$. Semua operasi ini digunakan dalam pengoperasian sub-blok 16-bit.

Algoritma ini melakukan iterasi yang terdiri dari atas 8 putaran dan I transformasi keluaran pada putaran ke 9. Berikut ini merupakan gambaran perhitungan komputasi dan transformasi keluaran.



Gambar 2. 2 Algoritma IDEA

Berdasarkan gambar diatas, terdapat beberapa operasi yang berbeda dan menjadi metodologi dari algoritma IDEA yakni perkalian modulo 2^{16} dan 16-bit integer, penjumlahan modulo 2^{16} dan 16-bit integer, serta operasi XOR 16-bit. Prinsip perancangan dari algoritma kriptografi IDEA adalah sebagai berikut :

1. *Diffusion*

   Konsep ini dilakukan dengan mengusahakan setiap bit dari setiap *plaintext* mempengaruhi setiap bit ciphertext dan setiap bit kunci untuk mempengaruhi setiap bit *ciphertext*.
2. *Confusion*

   Konsep ini dilakukan dengan mencampur tiga prinsip operasi grup aljabar yang berbeda-beda :
   a. XOR (Penambahan modulo 2)
   b. Penambahan modulo 2^{16}
   c. Perkalian dengan modulo $2^{16} + 1$

Menggunakan sebuah arsitektur serial-bit untuk melakukan perkalian modulo $2^{16} + 1$, implementasinya minimal membutuhkan perangkat keras yang sederhana. Arsitektur serial-bit memungkinkan algoritma menjadi lebih dalam untuk mendapatkan sebuah sistem dengan waktu clock sebesar 125MHz. Sebuah

implementasi pada Xilinx dapat dilakukan tes secara sukses, mendistribusikan sebuah keluaran sebesar 500Mb/sec. Dengan alat XCV1000-6, perkiraan performansi adalah 2.35Gb/detik, tiga lebih cepat dari urutan magnitude daripada sebuah implementasi perangkat lunak pada 450MHz Intel Pentium II. Desain ini cocok bagi aplikasi dengan enkripsi secara langsung (*online*) pada jaringan dengan kecepatan tinggi.

2.2.3.2 Proses Enkripsi IDEA

Pada proses enkripsi, algoritma IDEA ini ditunjukkan oleh gambar di atas, terdapat tiga operasi yang berbeda untuk pasangan sub-blok 16-bit yang digunakan, sebagai berikut :

1. XOR dua sub-blok 16-bit, bit per bit, yang disimbolkan dengan tanda $\oplus$
2. Penjumlahan integer modulo 2^{16} (mod 65.536) dua sub-blok 16-bit, dimana kedua sub-blok itu dianggap sebagai representasi biner dari integer biasa, yang disimbolkan dengan tanda $\boxplus$
3. Perkalian modulo ($2^{16} + 1$) yang nilainya 65.537 (bilangan ini prima dan invers dari perkalian bisa dijamin), dua sub-blok 16-bit, dimana kedua sub-blok 16-bit itu dianggap sebagai representasi biner dari integer biasa kecuali sub-blok nol dianggap mewakili integer 2^{16}, yang disimbolkan dengan tanda $\odot$

Setiap operasi dibuat pada blok 16-bit dan trik yang digunakan di sini adalah blok-blok tersebut yang nilainya adalah 0 (16-bit) akan digantikan oleh konstanta 2^{16} dari 17 bit. Membuktikan hal ini dengan jumlah digit yang sangat banyak tidak dapat dilakukan secara jelas. Walaupun begitu, kita dapat melihat contoh nya dengan menskalakannya dengan angka – angka yang kecil. Untuk contoh pada kelompok kecil n dimana $2^n + 1$ harus prima, n = 2 dimana $2^2 = 4$ dan ($2^2 + 1 = 5$). Penjelasan, Ingat bahwa 0 sama dengan $2^n = 4$ sehingga :

1. $0 \otimes 0 = 2^2 \times 2^2 = 16$
   $= 16 \bmod 5$
   $= 1$
2. $0 \otimes 0 = 2^2 \times 1 = 4$
   $= 4 \bmod 5$
   $= 4$
3. $0 \otimes 2 = 2^2 \times 2 = 8$
   $= 8 \bmod 5$
   $= 3$
4. $0 \otimes 3 = 2^2 \times 3 = 12$
   $= 12 \bmod 5$
   $= 2$

Operasi-operasi yang dilakukan antara lain :

1. $+ \bmod 2^n$ (mod 4)
2. $\otimes \bmod 2^n+1$ (mod 5)
3. XOR (mod 2)

4. Kalkulasi lain dengan nilai x dan y yang berbeda akan sama.

Penambahan Invers :

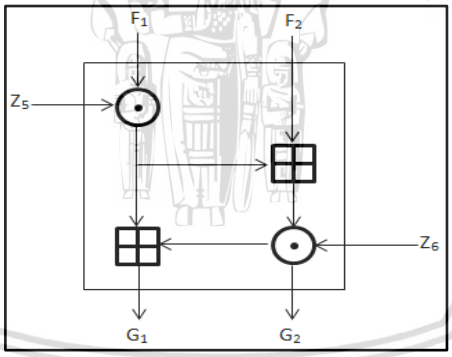
1. Modulo 2^{16} = Modulo 65536
2. A adalah penambahan invers dari B iff (A + B) mod 65536 = 0
3. Contoh penambahan pada 4 adalah: −4 mod 65536 ´ (−4 + 65536) = 65532

Perkalian Invers :

1. Modulo 2^{16} + 1 = Modulo 65537
2. A adalah perkalian invers dari B iff (A × B) mod 65537 = 1
3. Gunakan algoritma *Extended Euclidean* untuk mencari perkalian invers.

Gambar 2. 3 Simbol Operasi pada Algoritma IDEA

Gambar 2. 3 merupakan operasi-operasi yang mengakibatkan *confusion* dan dalam operasi-operasi ini tidak berlaku hukum distributif atau hukum asosiatif. Struktur dari *difusion* ini menciptakan sebuah blok dasar yang dinamakan MA *Structure Multiplication / Addition*. Ini hanya menggunakan dua kunci untuk setiap putarannya dan masukannya adalah F1, F2 sama seperti luarannya G1, G2 yang dihubungkan oleh XOR. Berikut ini adalah gambar dari blok utama IDEA :



Gambar 2. 4 Blok Utama IDEA

Tabel 2. 1 Operasi XOR, Penambahan, dan Perkalian

X	Y	X + Y	X ⊗ Y	X (xor) Y
00	00	00	01	00
00	01	01	00	01
00	10	10	11	10
00	11	11	10	11
01	00	01	00	01

01	01	10	01	00
01	10	11	10	11
01	11	00	11	10
10	00	10	11	10
10	01	11	10	11
10	10	00	00	00
10	11	01	01	01
11	00	11	10	11
11	01	00	11	10
11	10	01	01	01
11	11	10	00	00

Blok pesan terbuka dengan lebar 64-bit , X, dibagi menjadi 4 sub-blok 16-bit, X1, X2, X3, X4, sehingga X = (X1, X2, X3, X4). Keempat sub-blok 16-bit itu ditransformasikan menjadi sub-blok 16-bit, Y1, Y2, Y3, Y4, sebagai pesan rahasia 64-bit Y = (Y1, Y2, Y3, Y4) yang berada dibawah kendali 52 sub-blok kunci 16-bit yang dibentuk dari dari blok kunci 128 bit.

Keempat sub-blok 16-bit, X1, X2, X3, X4, digunakan sebagai masukan untuk putaran pertama dari algoritma IDEA. Dalam setiap putaran dilakukan operasi XOR, penjumlahan, perkalian antara dua sub-blok 16-bit dan diikuti pertukaran antara sub-blok 16-bit putaran kedua dan ketiga. Keluaran putaran sebelumnya menjadi masukan putaran berikutnya. Setelah putaran kedelapan dilakukan transformasi keluaran yang dikendalikan oleh 4 sub-blok kunci 16-bit.

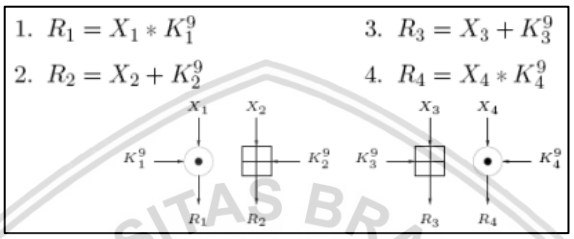
Pada setiap putaran dilakukan operasi-operasi berdasarkan gambar 2 sebagai berikut :

1. Perkalian X_1 dengan sub-kunci pertama
2. Penjumlahan X_2 dengan sub-kunci kedua
3. Pejumlahan X_3 dengan sub kunci ketiga
4. Perkalian X_4 dengan sub kunci keempat
5. Operasi XOR hasil langkah (1) dan (3)
6. Operasi XOR hasil angkah (2) dan (4)
7. Perkalian hasil langkah (5) dengan sub-kunci kelima
8. Penjumlahan hasil langkah (6) dengan langkah (7)
9. Perkalian hasil langkah (8) dengan subkunci keenam
10. Penjumlahan hasil langah (7) dengan (9)
11. Operasi XOR hasil langkah (1) dan (9)
12. Operasi XOR hasil langkah (3) dan (9)
13. Operasi XOR hasil langkah (2) dan (10)
14. Operasi XOR hasil langkah (4) dan (10)

Keluaran setiap putaran adalah 4 sub-blok yang dihasilkan pada langkah (11), (12), (13), dan (14) dan menjadi masukan putaran berikutnya. Setelah putaran kedelapan terdapat transformasi keluaran, yaitu :

1. Perkalian X_1 dengan sub-kunci pertama
2. Penjumlahan X_2 dengan sub-kuci ketiga
3. Penjumlahan X_3 dengan sub-kunci kedua
4. Perkalian X_4 dengan sub-kunci keempat

Terahir, keempat sub-blok 16-bit yang merupakan hasil operasi (1), (2), (3), dan (4) digabung kembali menjadi blok pesan rahasia 64-bit. Di bawah ini adalah gambar dari transformasi luaran.



Gambar 2. 5 Transformasi Luaran

Notasi superscript K^9_n menandakan jumlah putaran dari upa-kunci (*sub-key*). Pada kasus ini putaran sembilan adalah transformasi final.

2.2.3.3 Proses Dekripsi IDEA

Proses dekripsi menggunakan algoritma yang sama dengan proses enkripsi tetapi 52 buah subblok kunci yang digunakan masing-masing merupakan hasil turunan 52 buah sub-blok kunci enkripsi. Sebagai operasi yang dibentuk dari sebuah *field* terbatas, pada kasus ini akan diambil invers dari operasi XOR, penambahan oleh mod 2^{16} dan perkalian mod 2^{16} + 1, tergantung pada operasi yang dibuat pada fase chiper. Setiap upa-kunci (sub-key) adalah salah satu dari penambahan atau perkalian yang berkorespondensi dengan upa-kunci (*sub-key*) enkripsi. Invers:

1. Invers XOR: sama seperti fungsi yang sudah diaplikasikan.
2. Invers Penambahan: penambahan dengan mod 2^{16}.
3. Invers Perkalian: perkalian dengan mod 2^{16} + 1.

2.2.3.4 Pembentukan Upa-Kunci (Sub-key)

Sebanyak 52 upa-kunci blok 16-bit untuk proses enkripsi diperoleh dari sebuah kunci 128-bit pilihan pemakai. Enam upa-kunci digunakan setiap putarannya sehingga terpakai 48 upa-kunci. Sedangkan sisa empat upa-kunci untuk transformasi final. Blok kunci 128-bit dipartisi menjadi 8 sub-blok kunci 16-bit yang langsung dipakai sebagai 8 sub-blok kunci pertama. Kemudian blok kunci 128-bit dirotasi dari kiri 25 posisi untuk dipartisi lagi menjad 8 sub-blok kunci 16-bit berikutnya. Proses rotasi dan partisi itu diulangi lagi terus menerus sampai diperoleh 52 sub-blok kunci 16-bit.

Pada setiap operasi pada kunci 128 bit, 8 kunci dari 16 bits yang didapatkan, hanya enam kunci yang digunakan pada setiap putarannya. Kunci yang tersisa disimpan untuk putaran berikutnya. Di bawah ini adalah gambar pergeseran kunci pada IDEA :

Main key k = 128 bits

001 002 003 004 005 006 007 008 009 010 011 012 013 014 015 016 017 018 019 020 021 022 023 024 025 026 027 028 029 030 031 032

033 034 035 036 037 038 039 040 041 042 043 044 045 046 047 048 049 050 051 052 053 054 055 056 057 058 059 060 061 062 063 064

065 066 067 068 069 070 071 072 073 074 075 076 077 078 079 080 081 082 083 084 085 086 087 088 089 090 091 092 093 094 095 096

097 098 099 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128

	First 16 bits of key	....	Last 16 bits of key
1[a]	001 002 003 004 005 006 007 008 009 010 011 012 013 014 015 016	....	113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128
2[a]	026 027 028 029 030 031 032 033 034 035 036 037 038 039 040 041	....	010 011 012 013 014 015 016 017 018 019 020 021 022 023 024 025
3[a]	051 052 053 054 055 056 057 058 059 060 061 062 063 064 065 066	....	035 036 037 038 039 040 041 042 043 044 045 046 047 048 049 050
4[a]	076 077 078 079 080 081 082 083 084 085 086 087 088 089 090 091	....	060 061 062 063 064 065 066 067 068 069 070 071 072 073 074 075
5[a]	101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116	....	085 086 087 088 089 090 091 092 093 094 095 096 097 098 099 100
6[a]	126 127 128 001 002 003 004 005 006 007 008 009 010 011 012 013	....	110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125
7[a]	023 024 025 026 027 028 029 030 031 032 033 034 035 036 037 038	....	007 008 009 010 011 012 013 014 015 016 017 018 019 020 021 022

Gambar 2. 6 Pergeseran Kunci pada IDEA

Sedangkan penggunaan kunci setiap putarannya akan ditunjukkan oleh gambar di bawah ini :

First round :	$k_1k_2k_3k_4k_5k_6$	B[1….96]
Second round :	$k_7k_8k_9k_{10}k_{11}k_{12}$	B[97….128; 26….89]
Third round :	$k_{13}k_{14}k_{15}k_{16}k_{17}k_{18}$	B[90….128; 1….25; 51….8]
Fourth round :	$k_{19}k_{20}k_{21}k_{22}k_{23}k_{24}$	B[83….128; 1….50]
Fifth round :	$k_{25}k_{26}k_{27}k_{28}k_{29}k_{30}$	B[76….128; 1….43]
Sixth round :	$k_{31}k_{32}k_{33}k_{34}k_{35}k_{36}$	B[44….75; 101….128; 1…
Seventh round :	$k_{37}k_{38}k_{39}k_{49}k_{41}k_{42}$	B[37….100; 126….128; 1…
Eighth round :	$k_{43}k_{44}k_{45}k_{46}k_{47}k_{48}$	B[30….125]
Transformation :	$k_{49}k_{50}k_{51}k_{52}$	B[23….86]

Gambar 2. 7 Penggunaan Kunci setiap Putarannya

Distribusi dari bit-bit pada tiap upa-kunci pada setiap putarannya mengikuti seperti logika.

K_1 : 001 002 003 004 005 006 007 008 009 010 011 012 013 014 015 016

K_7 : 097 098 099 100 101 102 103 104 105 106 107 108 109 110 111 112

K_{13}: 090 091 092 093 094 095 096 097 098 099 100 101 102 103 104 105

K_{19}: 083 084 085 086 087 088 089 090 091 092 093 094 095 096 097 098

K_{23}: 076 077 078 079 080 081 082 083 084 085 086 087 088 089 090 091

K_{31}: 044 045 046 047 048 049 050 051 052 053 054 055 056 057 058 059

K_{37}: 037 038 039 040 041 042 043 044 045 046 047 048 049 050 051 052
K_{43}: 030 031 032 033 034 035 036 037 038 039 040 041 042 043 044 045

Gambar 2. 8 Kunci Pertama yang digunakan pada setiap Putaran

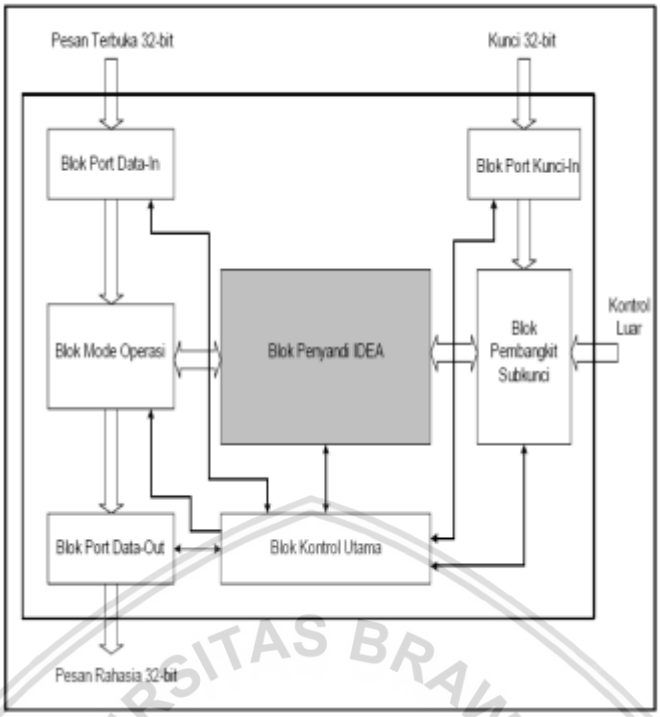
2.2.3.5 Arsitektur Umum Prosesor Kriptografi IDEA

Pada gambar berikut diperlihatkan arsitektur atau penggambaran umum sebuah processor yang mengolah sistem keamanan data dengan menggunakan algoritma IDEA. Jika kunci yang digunakan adalah "*IDEA es la clave*", cari 16 bit dari kunci kedua pada putaran keempat. Solusi :

01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
0	1	0	0	1	0	0	1	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	1
25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	1	1	0	0	1	0	1
49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72
0	1	1	1	0	0	1	1	0	0	1	0	0	0	0	0	0	1	1	0	1	1	0	0
73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96
0	1	1	0	0	0	0	1	0	0	1	0	0	0	0	0	0	1	1	0	0	0	1	1
97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114						
0	1	1	0	1	1	0	0	0	1	1	0	0	0	0	1	0	1						
115	116	117	118	119	120	121	122	123	124	125	126	127	128										
1	1	0	1	1	0	0	1	1	0	0	1	0	1										

Gambar 2. 9 Upa-Kunci

Seperti pada setiap putaran enam penggunaan upa-kunci (*subkeys*), kunci kedua dari putaran keempat akan berupa penjumlahan 3∗6+2 = 20. Seperti pada kunci 19 yang berhenti pada bit ke 98, kunci 20 akan sama 16 bit, hal ini berasal dari 99 ke 114: k20 = 10110001 10000101.



Gambar 2. 10 Arsitektur Prosesor IDEA

Keterangan :

1. Blok penyandi IDEA.
   Blok ini berfungsi untuk melakukan proses penyandian data. Jika sub-kunci yang diproses oleh blok ini berupa sub-kunci enkripsi maka pesan yang dihasilkan adalah pesan rahasia (*chipertext*) dan jika yang diproses berupa subkunci dekripsi maka pesan yang dihasilkan adalah pesan sebenarnya (*plaintext*).
2. Blok pembangkit sub-kunci.
   Blok ini berfungsi untuk membentuk 52 buah sub-kunci enkripsi 16 bit dari kunci enkripsi 128 bit. Sehingga membentuk 52 buah subkunci dekripsi 16 bit dari kunci dekripsi 128 bit.
3. Blok port *data-in*
   Blok ini berfungsi untuk membaca 2 buah blok data masukan 32 bit dan penyimpananya sebagai blok data masukan 64 bit yang akan dienkripsi atau didekripsi.
4. Blok port *data-out*
   Blok ini berfungsi untuk mengeluarkan blok data keluaran 64 bit yang merupakan hasil enkripsi atau dekripsi dengan cara membagi menjadi 2 buah blok data keluaran 32 bit.
5. Blok port kunci-n
   Blok ini berfungsi untuk membaca 4 buah blok kunci 32 bit dan menyimpanya sebagai blok kunci 128 bit.
6. Blok mode operasi
   Blok ini berfungsi untuk menentukan mode operasi yang digunakan pada proses ekripsi dan dekripsi.

7. Blok kontrol
   Blok ini berfungsi untuk mengontrol operasi antara blok fungsional yang menyusun sebuah blok besar seperti sinkronisasi transfer data antar blok.

2.2.4 VHSIC Hardware Description Language (VHDL)

VHDL merupakan salah satu bahasa yang digunakan untuk mendeskripsikan suatu perangkat keras. VHDL merupakan kependekan dari *VHSIC Hardware Description Language*, sedangkan VHSIC kependekan dari *Very High Speed Integrated Circuit*.

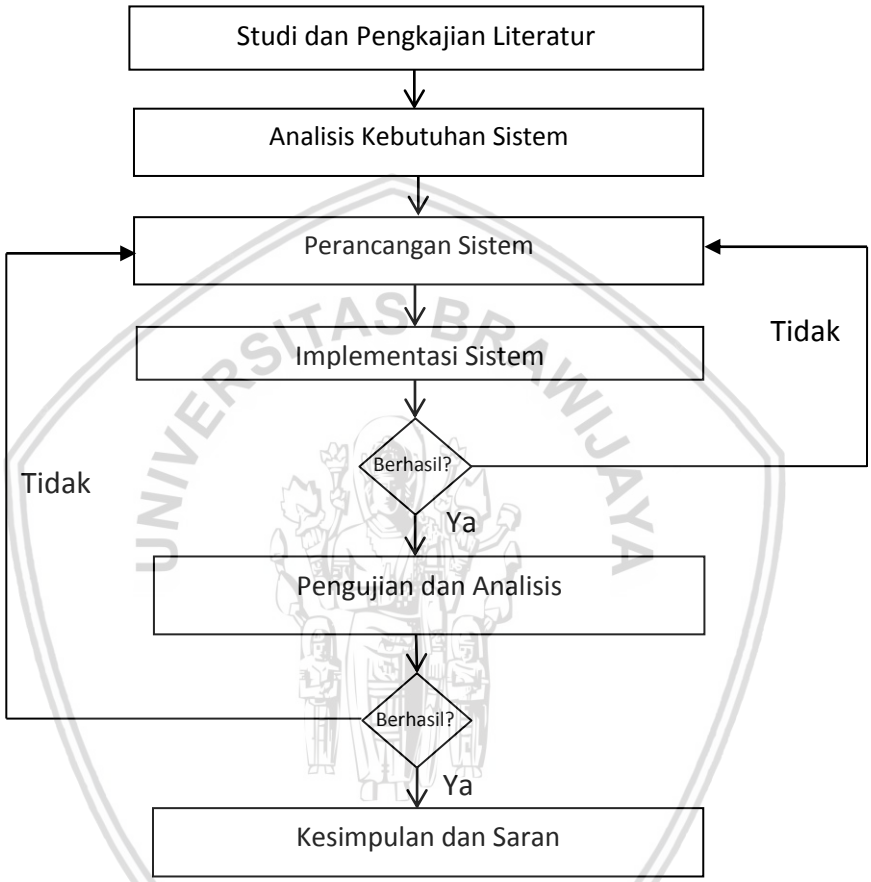
Pada awalnya VHDL merupakan standar dokumentasi perangkat keras yang disponsori oleh Departemen Pertahanan Amerika Serikat sekitar tahun 1980, yang kemudian dokumentasi tersebut dikirimkan ke *Institute of Electrical and Electronic Engineers (IEEE)* dan diratifikasi oleh IEEE dengan *standard* IEEE1076 dan IEEE 1164 dengan nama VHDL-87. Setelah peluncuran pertama, berbagai varian VHDL dikembangkan untuk memfasilitasi berbagai macam kebutuhan perancangan dan pemodelan perangkat keras (Tolstrup, Nielson, & Nielson, 2007).

2.2.4.1 Struktur Dasar

Secara umum, struktur bahasa pemrograman VHDL terdiri dari tiga bagian pokok, yaitu deklarasi *Library, Entity* serta *Architecture*. Deskripsi *library* merupakan kumpulan potongan kode yang sering digunakan, sebuah entity berisi deklarasi Input/Output dari suatu sistem, sedangkan deklarasi *architecture* berisi deskripsi dari suatu rangkaian atau fungsi logika yang akan diimplementasikan. Di dalam *architecture* dapat menangani baik rangkaian sekuensial maupun rangkaian kombinasional dan dalam 1 (satu) modul VHDL dimungkinkan untuk menggunakan lebih dari 1 (satu) *architecture*.

BAB 3 METODOLOGI

Bab ini menjelaskan langkah-langkah yang akan ditempuh dalam penyusunan skripsi, meliputi studi dan pengkajian literatur, analisis kebutuhan sistem, perancangan sistem, implementasi sistem, pengujian dan analisis, kesimpulan dan saran.



Gambar 3. 1 *Flowchart* Metode Penelitian

3.1 Studi Literatur

Dalam perancangan dan implementasi penelitian ini, perlu diadakan studi literatur. Literatur digunakan sebagai teori penguat dan landasan dasar dalam penelitian. Teori pendukung tersebut di dapat dari buku, jurnal, paper dan internet. Literatur yang digunakan meliputi :

1. Kriptografi
   a. Jenis-jenis algoritma kriptografi
   b. *Stream cipher*
   c. *Block cipher*

2. Algoritma *International Data Encryption Algorithm*
   a. XOR
   b. Penjumlahan integer modulo (2^{16})
   c. Perkalian modulo ($2^{16} + 1$)
   d. Sub-key
3. *Embedded System*
   a. FPGA
   b. FPGA Xilinx Spartan 3-e
   c. Pin Out FPGA Xilinx Spartan 3-e
   d. RS232

3.2 Analisis Kebutuhan Sistem

Analisis Kebutuhan bertujuan untuk mendapatkan semua kebutuhan yang diperlukan oleh sistem yang akan dibangun dan di uji. Analisis kebutuhan dilakukan dengan mengidentifikasi kebutuhan sistem. Analisis kebutuhan tersebut meliputi kebutuhan perangkat lunak dan kebutuhan perangkat keras.

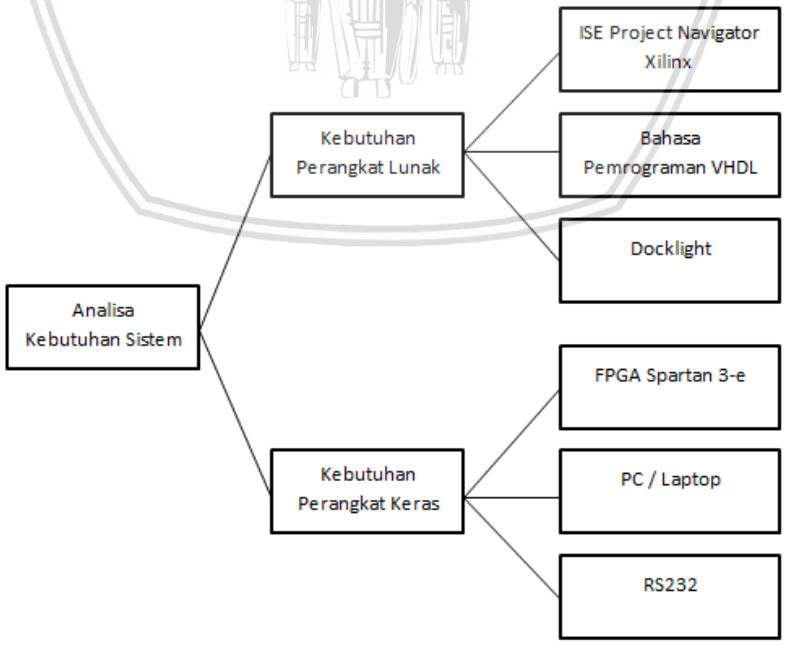
Sesuai dengan tujuan dari penulisan ini yang mengarah ke *embedded system* yang dimana suatu sistem berbasis mikroprosesor yang dibangun untuk mengendalikan satu atau beberapa fungsi dan tidak dirancang untuk dapat di program oleh pengguna akhir seperti komputer pribadi maka dibutuhkan kebutuhan perangkat yang sesuai. Kebutuhan perangkat pada penelitian ini adalah pada penelitian ini dibutuhkan *embedded system* yang dapat bekerja secara cepat dan efisien dalam mengolah algoritma IDEA yang cukup rumit. Dari penelitian-penelitian sebelumnya muncul nama FPGA yang bekerja lebih efisien dan lebih berperforma dibandingkan CPU dan GPU, selain itu FPGA jika dibandingkan dengan *embedded system* yang lainnya misalnya Rasphberry atau Arduino, FPGA berjalan pada *layer physical* sehingga proses eksekusinya lebih cepat sedangkan Rasphberry atau Arduino berjalan pada layer aplikasi. Dari pertimbangan tersebut maka dipilih FPGA untuk menerapkan penelitian ini.

Pada penelitian ini dibutuhkan juga Docklight untuk melakukan komunikasi serial yang menghubungkan antara PC dan FPGA dengan menggunakan Modul RS232 sebagai penghubung. Nantinya pada Docklight tersebut dapat menampilkan hasil dari enkripsi data teks, juga dibutuhkan *keyboard* yang nantinya berfungsi sebagai masukan data *plaintext*. Dari pertimbangan tersebut maka dipilih FPGA Xilinx Spartan 3-e untuk menjadi board implementasi pada penelitian ini dikarenakan FPGA tersebut terdapat fitur seperti mempunyai 500 ribu sistem gerbang dengan 10.476 logika sel dengan performa yang tinggi sebagai solusi dari kompleksitas algoritma IDEA, mempunyai *multi-voltage* dan *multi-standard SelectIO* masing-masingnya sebanyak 376 I/O pin dan 156 pasang

sinyal diferensial dan mempunyai dua *port serial* RS232 sebagai fitur yang mendukung komunikasi yang dilakukan secara serial.

Pada penelitian ini untuk merancang system dibutuhkan sebuah PC ataupun laptop. Kebutuhan perangkat lunak digunakan untuk merancang program enkripsi data text dengan algoritma IDEA. Untuk mendesain dan merancang sistem dibutuhkan *compiler* yang cocok dengan hardware yang digunakan yaitu Xilinx Spartan 3-e. Xilinx Spartan 3-e memiliki software bawaan yang sudah pasti cocok dengan FPGA tersebut dan nantinya juga bisa digunakan untuk mengimportkan program ke dalam FPGA. Maka dari itu dipilih ISE Project Navigator sebagai *compiler* dan sebagai desain program system. Pada kebutuhan perangkat lunak yang lainnya adalah bahasa pemrograman. Pada penelitian ini dibutuhkan bahasa pemrograman yang dapat merancang sistem digital yang kompleks.

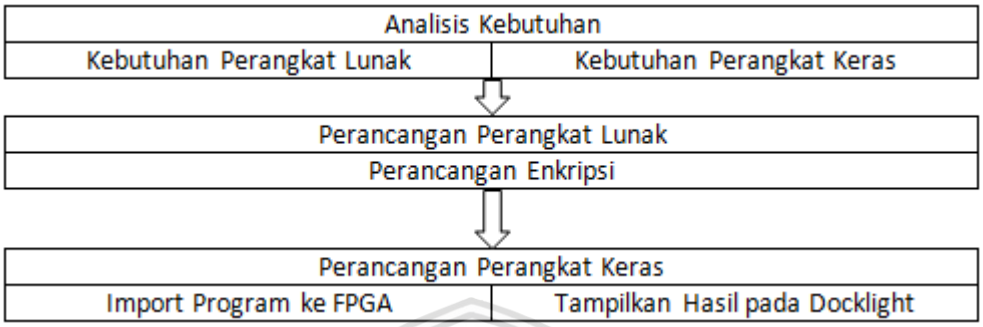
Pada penelitian ini juga dibutuhkan bahasa pemrograman yang dapat menggunakan gabungan level dari model yang memiliki arsitektur yang berbeda. *System* yang dibuat nantinya memiliki *multiple behavior* dalam sebuah *interface*. Selain itu model tersebut memungkinkan terjadi pertukaran dan implementasi multiple secara terus menerus. Dilihat dari kebutuhan bahasa pemrograman tersebut maka VHDL yang cocok untuk kebutuhan tersebut, karena karakteristik VHDL adalah mempunyai kemampuan yang lebih untuk merancang sistem digital yang kompleks. Satu hal penting tentang keunggulan VHDL dibandingkan verilog adalah kemampuannya untuk menggunakan gabungan level dari model yang memiliki arsitektur yang berbeda. Dari kebutuhan-kebutuhan tersebut maka dirancanglah diagram kebutuhan sistem seperti yang ditunjukkan pada gambar 3.2.



Gambar 3. 2 Diagram Kebutuhan Sistem

3.3 Perancangan Sistem

Perancangan sistem dilakukan setelah semua kebutuhan sistem didapatkan melalui tahap analisis kebutuhan yang meliputi kebutuhan perangkat lunak dan kebutuhan perangkat keras. Seperti yang ditunjukkan dalam Gambar 3.3.

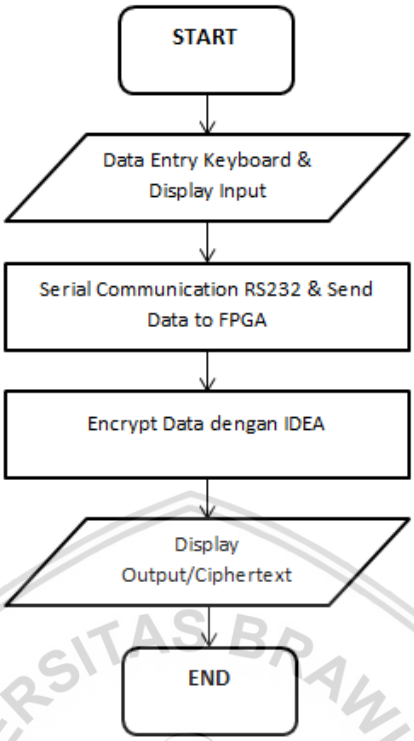


Gambar 3. 3 Alur Perancangan Sistem Secara Umum

Pada perancangan enkripsi dilakukan dengan algoritma *International Data Encryption Algorithm* (IDEA). Algoritma IDEA merupakan algoritma simetris yang beroperasi pada sebuah blok pesan *plaintext* terbuka dengan panjang blok 64-bit. Memilih panjang blok sebanyak 64 bit karena dirasa sudah cukup panjang bila kita akan melakukan analisis statistikal. Menggunakan kunci yang sama berukuran 128-bit, untuk proses enkripsi maupun dekripsi. Kunci yang digunakan panjangnya 128 bit karena bila dilakukan serangan *brute force* pada IDEA menggunakan sebuah sistem yang dapat melakukan pengujian 109 kunci per detiknya, akan diperlukan waktu 1013 tahun untuk mencari kuncinya. Pesan rahasia yang dihasilkan oleh algoritma ini (*chipertext*) berupa blok pesan rahasia dengan lebar satu blok 64-bit. Sebuah sistem yang dapat melakukan pengujian 109 kunci akan mencari semua kemungkinan dari 256 kunci DES dalam waktu di bawah satu detik (72 millidetik). Pada perancangan perangkat keras terdapat proses *import* program ke FPGA dan ditampilkan dalam komunikasi serial Docklight dengan bantuan kabel serial RS232 pada proses tersebut digunakan ISE Project Navigator sebagai perantara atau penghubung dalam setiap proses tersebut.

3.3.1 Perancangan Enkripsi

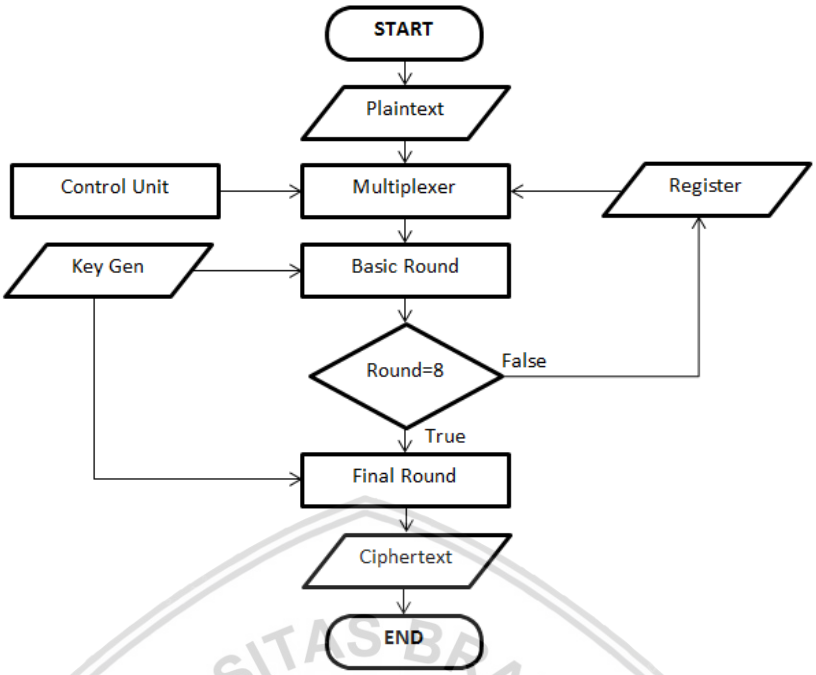
Perancangan enkripsi merupakan bagian perancangan perangkat lunak. Enkripsi data teks dilakukan dengan algoritma IDEA. Diagram alir dari proses umum system dijelaskan pada gambar 3.4.



Gambar 3. 4 Diagram alir proses umum sistem

3.3.1.1 Proses Encrypt Data dengan IDEA

Proses enkripsi dilakukan dengan algoritma IDEA. Pada proses enkripsi, algoritma IDEA ini ditunjukkan oleh gambar di atas, terdapat tiga operasi yang berbeda untuk pasangan sub-blok 16-bit yang digunakan. Setiap operasi dibuat pada blok 16-bit dan trik yang digunakan di sini adalah blok-blok tersebut yang nilainya adalah 0 (16-bit) akan digantikan oleh konstanta 2^{16} dari 17 bit. Membuktikan hal ini dengan jumlah digit yang sangat banyak tidak dapat dilakukan secara jelas. Walaupun begitu, kita dapat melihat contoh nya dengan menskalakannya dengan angka-angka yang kecil. Untuk contoh pada kelompok kecil n dimana $2^n + 1$ harus prima, n = 2 dimana $2^2 = 4$ dan ($2^2 + 1 = 5$). Flowchart proses enkripsi akan dijelaskan pada Gambar 4.2.



Gambar 3. 5 *Flowchart* Enkripsi IDEA

Pada Gambar 3.5 menjelaskan proses enkripsi algoritma IDEA dengan plaintext 64-bit yang dibagi menjadi 4 bagian dimana dapat dioperasikan nantinya, key gen 128-bit yang dibagi menjadi 8 bagian (*sub keys*) dengan melakukan *shifting* sebanyak 25-bit ke LSB sehingga mendapatkan 52 variasi *sub key* dan akhirnya akan mendapatkan 64-bit *ciphertext* dari algoritma IDEA tersebut. Algoritma IDEA dapat dijelaskan pada tabel 3.1.

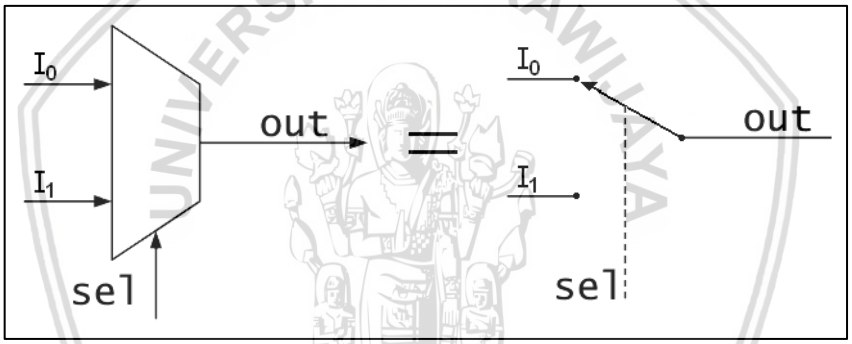
Tabel 3. 1 *Pseudocode* Enkripsi IDEA

```
Nama algoritma : Algoritma Enkripsi IDEA
Input : Plaintext, Key {P 64-bit, K 128-bit}
Output : Ciphertext {C 64-bit}
Proses :
(Round, Key) ← Ekspansi Kunci {52 Subkey diambil dari kunci utama yang sudah di shifting dan dibutuhkan 6 subkeys untuk masing-masing round. Untuk final round hanya dibutuhkan 4 subkeys.}
Control Unit [] = {sStart, sReady, sWorking}
for i = 1 → Round do
If Register = 0 then
          Multiplexer (Plaintext, Control Unit)
          Basic Round (Multiplexer, Key)
          Basic Round = Register
else
```

```
            Multiplexer (Register, Control Unit)
            Basic Round (Multiplexer, Key)
            Basic Round = Register
End if
End for
Final Round (Register, Key)
Ciphertext = Final Round
```

3.3.1.2 Multiplexer

Proses *multiplexer* digunakan untuk melakukan seleksi pada masukan 16-bit plaintext ataupun nilai pada 16-bit register dengan *control unit*, artinya ketika *control unit* memberikan sinyal S atau start telah bernilai 0 maka masukan dapat diproses serta ketika sinyal S atau start belum ada perubahan dari nilai awal (*default* S=1) maka keluaran tersebut akan di set 'X' karena masukan tidak dapat diproses. Berikut adalah skema dari *multiplexer* tersebut.

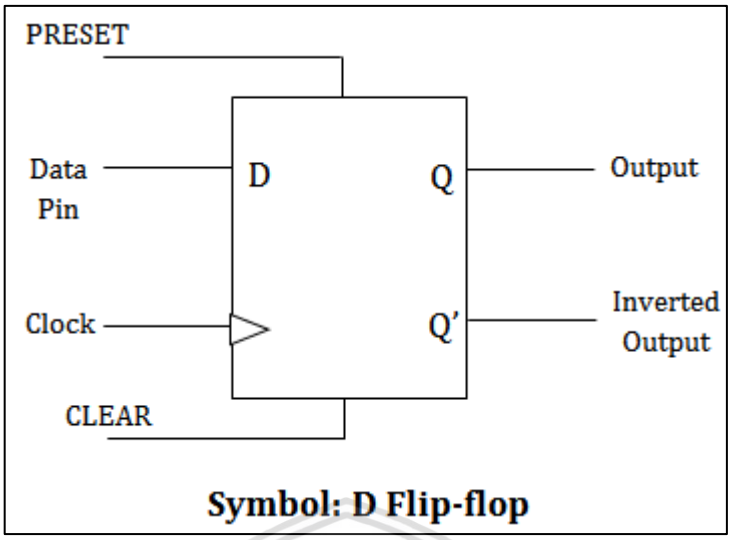


Gambar 3. 6 Skema *Multiplexer*

Berdasarkan gambar 3.6 diatas, *multiplexer* akan berfungsi sebagai selektor untuk masukan beberapa data, sehingga dalam penulisan ini terdapat dua masukan 16-bit yang akan di seleksi untuk dapat menjadi satu masukan 16-bit pada proses *basic round*.

3.3.1.3 Register

Proses *register* digunakan sebagai blok penyimpanan sementara yang mana *register* tersebut akan menyimpan hasil dari *basic round* dan akan menjadi masukan kembali sesuai dengan jumlah *round* yang tersedia. berdasarkan gambar 4.2 ketika jumlah *round* masih belum sama dengan delapan maka nilai keluaran dari *basic round* sementara akan disimpan di blok *register* dan ketika jumlah *round* pada proses tersebut sudah mencapai delapan maka nilai keluaran akan menjadi masukan pada *final round*, tidak akan disimpan kembali pada *register*. Berikut adalah skema dari *register* yang digunakan.



Gambar 3. 7 Skema *D Flip-flop*

Berdasarkan gambar 3.7 diatas merupakan sistem spesifik yang digunakan pada *register* algoritma IDEA. Dapat dikatakan pula bahwa *register* merupakan sekumpulan *flip-flop* yang dapat dipakai untuk menyimpan dan untuk mengolah informasi dalam bentuk linier. Pada penelitian ini, *d flip-flop* digunakan sebagai *flip-flop* pengunci data, artinya pada *flip-flop* tersebut jika terdapat berapapun nilai yang diberikan pada *input* D maka akan dikeluarkan dengan nilai yang sama pada *output* Q. Dari hal tersebut, *d flip-flop* sangat tepat bila diaplikasikan pada rangkaian-rangkaian yang memerlukan penyimpanan data sementara sebelum diproses berikutnya.

3.3.1.4 Control Unit

Proses *control unit* merupakan salah satu elemen penting karena proses tersebut akan melakukan pengawasan terhadap masukan data atau informasi yang ada. Pada *control unit* yang dimaksud pada perancangan ini terdapat beberapa variabel pendukung yakni variabel *Ready, Starting* dan *Working*. Variabel-variabel tersebut dimaksudkan untuk melakukan pengawasan terhadap proses-proses yang ada pada algoritma IDEA seperti pada proses *multiplexer* yang menggunakan *control unit* sebagai selektor daripada *multiplexer* dan sebagainya.

3.3.1.5 Key Generator

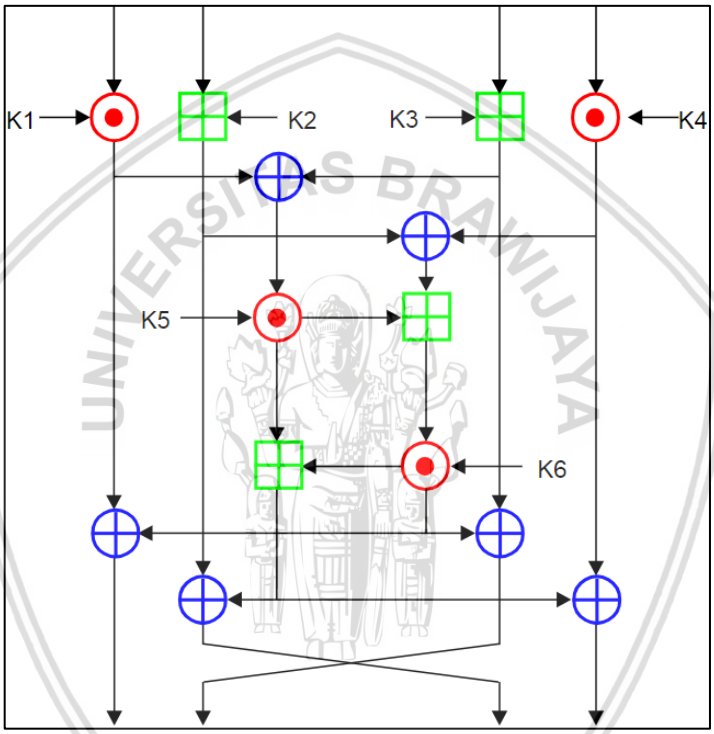
Proses *key generator* atau dapat dikatakan sebagai proses pembuatan kunci pada algoritma IDEA merupakan hal yang penting dalam alur enkripsi data. Pembuatan kunci maupun upa-kunci dilakukan untuk memenuhi konsep dasar pada algoritma IDEA yang mana dibutuhkan 52 *sub-keys* serta akan disebar pada seluruh round. 52 *sub-keys* tersebut dibutuhkan pada algoritma IDEA dan dibagi untuk delapan *round* pertama dengan pembagian enam *sub-keys* dan empat *sub keys* terakhir untuk *final round*. Berikut adalah *flowchart* dari pembentukan *sub-keys* algoritma IDEA.

Gambar 3. 8 *Flowchart Key Generate* Algoritma IDEA

Berdasarkan gambar 3.8 diatas, dijelaskan bahwa pembentukan 128-bit kunci utama hingga didapatkan 52 upa-kunci yang nantinya dapat dioperasikan pada delapan *round* dan satu *round* algoritma IDEA. Dimulai dari deklarasi 128-bit kunci utama yang mana kunci tersebut akan dibagi menjadi 8 bagian yang masing-masingnya terdapat 16-bit, dengan demikian terdapat deklarasi upa-kunci K_1, K_2, K_3, K_4, K_5, K_6, K_7, dan K_8. Kemudian, ketika seluruh kunci utama telah digunakan, maka kunci utama tersebut akan diputar ke kiri sebanyak 25 bit. Sehingga kunci utama tersebut akan terganti dengan 128-bit kunci yang baru yang mana kunci tersebut akan dibagi lagi menjadi delapan bagian dan begitu selanjutnya hingga terbentuk 52 upa-kunci. Namun, ke delapan upa-kunci tersebut tidak dipakai sekaligus melainkan masing-masing *round* hanya menggunakan enam upa-kunci. Maka dari pernyataan tersebut didapat bahwa untuk round pertama menggunakan K_1, K_2, K_3, K_4, K_5, K_6 dan untuk *round* kedua menggunakan K_7, K_8, K_9, K_{10}, K_{11}, K_{12} dan begitu seterusnya hingga pada *round* terakhir menggunakan K_{48}, K_{49}, K_{50}, K_{51}.

3.3.1.6 Basic Round

Proses *basic round* yang terdapat pada algoritma IDEA merupakan hal yang paling utama dari proses enkripsi algoritma tersebut. *Basic round* memiliki beberapa operasi aljabar yang berbeda, dimana operasi-operasi tersebut dilakukan untuk mendapatkan suatu keluaran enkripsi. Operasi-operasi yang terdapat pada *basic round* diataranya, operasi XOR ($\oplus$), operasi penjumlahan modulo 2^{16} ($\boxplus$), dan operasi perkalian modulo (2^{16} + 1) ($\odot$). Seluruh operasi tersebut dilakukan sebanyak delapan putaran. *Basic round* tersebut masing-masing putarannya menggunakan 6 upa-kunci, maka 48 dari 52 upa-kunci akan digunakan pada *basic round* tersebut. Berikut adalah alur dari *basic round* algoritma IDEA.



Gambar 3. 9 Proses *Basic Round* Enkripsi Algoritma IDEA

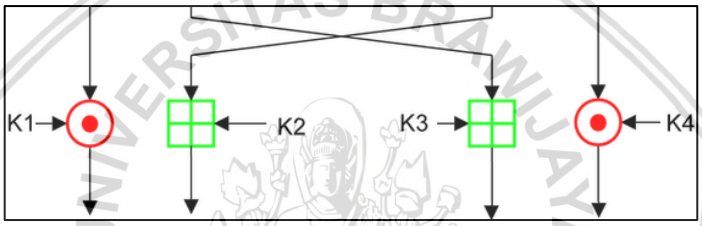
Berdasarkan gambar 3.9 diatas, proses *basic round* akan melakukan 14 operasi aritmatika yang mana seluruh operasi pada gambar diatas dihitung sebagai satu proses *basic round* dan dibutuhkan delapan putaran untuk menyelesaikan *basic round* algoritma IDEA. Berikut ini adalah penjelasan spesifik proses *basic round*.

1. Perkalian X_1 dengan sub-kunci pertama
2. Penjumlahan X_2 dengan sub-kunci kedua
3. Pejumlahan X_3 dengan sub kunci ketiga
4. Perkalian X_4 dengan sub kunci keempat
5. Operasi XOR hasil langkah (1) dan (3)
6. Operasi XOR hasil angkah (2) dan (4)
7. Perkalian hasil langkah (5) dengan sub-kunci kelima
8. Penjumlahan hasil langkah (6) dengan langkah (7)

9. Perkalian hasil langkah (8) dengan subkunci keenam
10. Penjumlahan hasil langah (7) dengan (9)
11. Operasi XOR hasil langkah (1) dan (9)
12. Operasi XOR hasil langkah (3) dan (9)
13. Operasi XOR hasil langkah (2) dan (10)
14. Operasi XOR hasil langkah (4) dan (10)

3.3.1.7 Final Round

Proses *final round* yang terdapat pada algoritma IDEA merupakan kelanjutan daripada *basic round*. Maksudnya adalah ketika *basic round* telah melakukan delapan putaran maka tahap selanjutnya adalah proses *final round*. *Final round* pada algoritma IDEA hanya dilakukan sekali proses atau dapat dikatakan tanpa perulangan proses, maka upa-kunci yang digunakan pada proses ini pun hanya dibutuhkan empat upa-kunci. Artinya setelah 48 upa-kunci yang telah digunakan pada *basic round*, maka bersisa empat upa-kunci terakhir untuk digunakan pada *final round* algoritma IDEA. Berikut adalah alur dari *final round* algoritma IDEA.



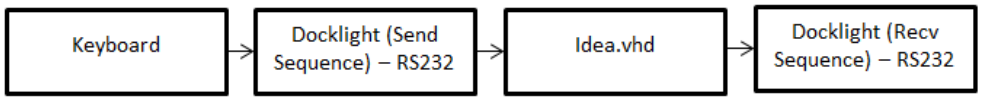
Gambar 3. 10 Proses *Final Round* Enkripsi Algoritma IDEA

Berdasarkan gambar 3.10 diatas, proses final round akan melakukan empat operasi aritmatika. Proses tersebut akan dilakukan sehingga akan mendapatkan keluaran sebagai *ciphertext* algoritma IDEA. Berikut adalah penjelasan spesifik proses final round.

1. Perkalian X_1 dengan sub-kunci pertama
2. Penjumlahan X_2 dengan sub-kuci ketiga
3. Penjumlahan X_3 dengan sub-kunci kedua
4. Perkalian X_4 dengan sub-kunci keempat

3.3.2 Perancangan *Input* dan *Output* Program

Perancangan input dan output program akan ditunjukkan pada gambar berikut ini.



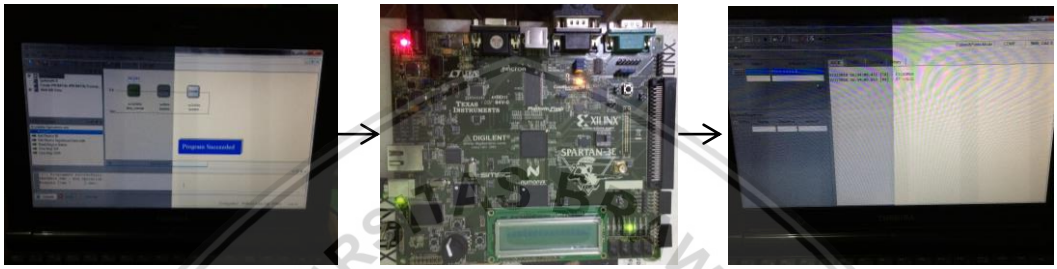
Gambar 3. 11 Perancangan *Input* dan *Output* Program

Berdasarkan gambar 4.8 diatas, terdapat masukan program akan dilakukan pada keyboard yang mana dapat memasukkan format teks (ASCII), hex, *decimal* ataupun *binary* 64-bit. Kemudian digunakan aplikasi tambahan Docklight dan kabel serial RS232 untuk menterjemahkan masukan *keyboard* sekaligus

melakukan komunikasi serial antar PC dan FPGA tersebut. Lalu idea.vhd akan di implementasikan pada FPGA sehingga dapat mengoperasikan enkripsi algoritma IDEA dengan masukan dari keyboard sebelumnya. Selanjutnya hasil dari enkripsi algoritma IDEA dapat ditampilkan dalam bentuk teks (ASCII), hex, decimal ataupun binary 64-bit pada Docklight dengan bantuan kabel serial RS232.

3.3.3 Perancangan Perangkat Keras

Berdasarkan diagram kebutuhan sistem pada bab sebelumnya, perancangan perangkat keras dilakukan setelah analisis kebutuhan dan perancangan perangkat lunak telah dilakukan. Perancangan perangkat keras dapat dijelaskan pada gambar berikut ini.



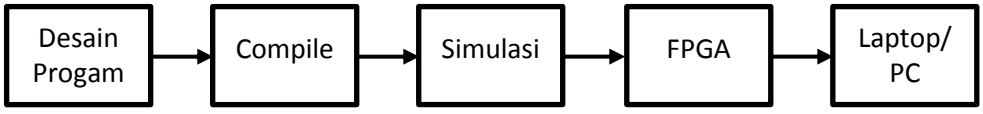
Gambar 3. 12 Perancangan Perangkat Keras

Perancangan perangkat keras dilakukan setelah perancangan perangkat lunak selesai dilakukan. Seperti yang sudah disebutkan didalam analisis kebutuhan perangkat keras, pada perancangan perangkat keras dibutuhkan Laptop/PC sebagai pengolah masukan dan keluaran hasil dari pemrosesan, FPGA Xilinx Spartan 3-e sebagai pengolah algoritma IDEA, dan RS232 sebagai komunikator serial yang mana menghubungkan antara Laptop/PC dengan FPGA Xilinx Spartan 3-e.

Setelah desain program selesai dan program sukses langkah selanjutnya adalah mengimportkan program kedalam FPGA Xilinx Spartan 3-e, pada proses ini dibutuhkan kabel USB untuk menghubungkan antara Laptop/PC dengan FPGA Xilinx Spartan 3-e. Setelah program sukses di*import*kan di dalam papan FPGA.

3.4 Implementasi Sistem

Implementasi merupakan penjelasan mulai dari perancangan program sampai hasil akhir yang ditampilkan dalam Laptop/PC. Seperti yang di gambarkan dalam Gambar 3.13.



Gambar 3. 13 Gambaran Secara Umum Implementasi Enkripsi Data Text dengan Algoritma IDEA

Desain program, *compile* sampai simulasi di jalankan di perangkat lunak pendukung bawaan dari Xilinx. Jika program sukses, *import* program dalam papan FPGA yang kemudian hasilnya akan ditampikan dalam Monitor dengan kabel RS232. Proses ini mungkin tidak berjalan satu kali, jika dirasa program belum sesuai kebutuhan, program akan di desain kembali dalam perangkat lunak.

3.5 Pengujian dan Analisis

Pengujian dilakukan dengan 3 tahap :

1. Pengujian fungsional
2. Pengujian validitas
3. Pengujian waktu eksekusi

Pengujian yang pertama adalah pengujian fungsional. Pengujian ini dilakukan untuk mengetahui sistem dapat berjalan dengan baik berdasarkan perancangan yang ada. Pengujian tersebut dilakukan dengan cara menghubungkan seluruh rancangan sistem dan membandingkan masukan dan keluaran yang sesuai.

Pengujian kedua adalah pengujian validitas. Pengujian ini dilakukan dengan cara membandingkan hasil *ciphertext* yang terdapat pada referensi hasil dari *vector test* dengan kunci yang sama. Jika memang sistem memberikan hasil yang valid maka seharusnya hasil yang dikeluarkan akan sama dengan hasil yang terdapat pada referensi hasil dari *vector test*. Dari pengujian ini akan diambil 10 data yang akan di uji.

Pengujian ketiga adalah pengujian waktu. Pengujian ini dilakukan untuk mengetahui waktu eksekusi yang dibutuhkan sistem untuk mengenkripsi data. Pengujian ini akan dilakukan dengan cara memasukkan beberapa data sampel yang berbeda. Hasil dari setiap data sampel akan dibandingkan dan di analisa.

3.6 Kesimpulan

Pengambilan kesimpulan dilakukan setelah semua tahapan perancangan, implementasi, pengujian dan analisis sistem telah selesai dilakukan. Kesimpulan diambil dari hasil pengujian dan analisis terhadap sistem yang dibuat. Kesimpulan di ambil untuk mengetahui validitas algortima IDEA.

Tahap akhir yang dilakukan adalah saran untuk memperbaiki kesalahan – kesalahan yang terjadi dalam pembuatan enkripsi teks dengan algoritma IDEA pada FPGA dan menyempurnakan penulisan serta untuk memberikan pertimbangan atas pengembangan aplikasi selanjutnya.

BAB 4 IMPLEMENTASI SISTEM

4.1 Lingkungan Implementasi

Lingkungan implementasi dari enkripsi teks dengan algoritma IDEA pada FPGA Xilinx Spartan 3-e meliputi lingkungan perangkat lunak (*software*) dan lingkungan perangkat keras (*hardware*).

4.1.1 Lingkungan Perangkat Lunak

Lingkungan perangkat lunak yang digunakan dalam mengimplementasikan enkripsi teks dengan algoritma IDEA adalah sebagai berikut.

1. Sistem Operasi Windows 7 Ultimate 32-bit sebagai proses implementasi
2. ISE Project Navigator 13.4 merupakan *software development* bawaan dari Xilinx Spartan 3-e yang digunakan untuk pemrograman.
3. Docklight sebagai serial komunikasi.

4.1.2 Lingkungan Perangkat Keras

Lingkungan perangkat keras yang digunakan dalam mengimplementasikan enkripsi teks dengan algoritma IDEA pada FPGA adalah sebagai berikut.

1. FPGA Xilinx Spartan 3-e
2. Kabel USB
3. Kabel serial RS232
4. Laptop prosesor Intel® Atom™ CPU N455
5. RAM 1.00 GB
6. *Keyboard*

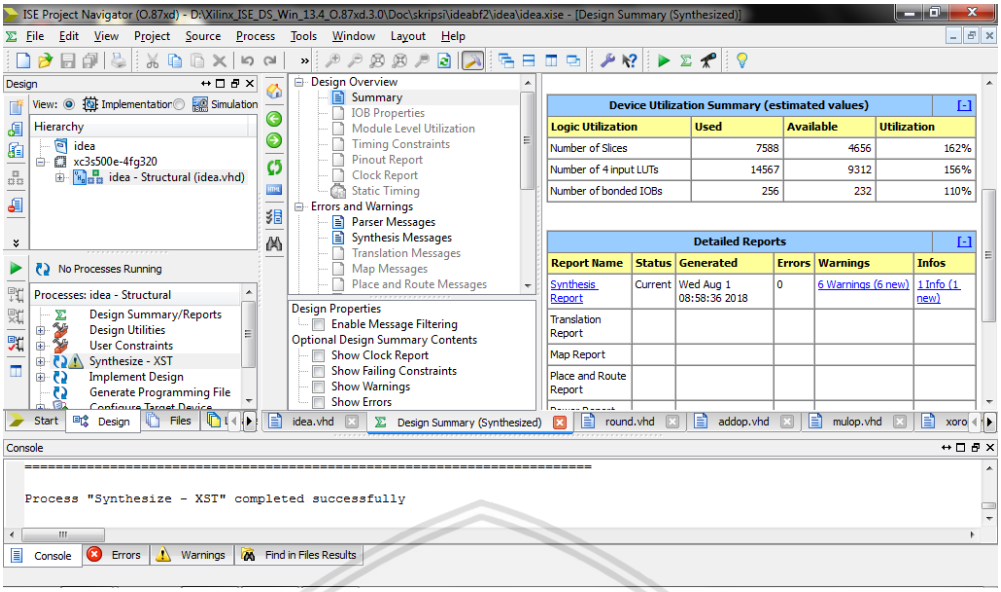
4.2 Batasan Implementasi

Beberapa batasan implementasi dalam implementasi teks dengan algoritma IDEA pada FPGA sebagai berikut.

1. Enkripsi data text di implementasikan di papan FPGA Xilinx Spartan 3-e.
2. Implementasi dititik beratkan pada proses Enkripsi.
3. Panjang karakter 8 karakter untuk *plaintext* dan 8 karakter yang ditampilkan pada *ciphertext*.
4. Algoritma enkripsi yang digunakan adalah algoritma IDEA.

4.3 Implementasi Perangkat Lunak

Implementasi enkripsi teks dengan algoritma IDEA pada FPGA berdasarkan perancangan sistem pada bab sebelumnya. Pada implementasi perangkat lunak yaitu implementasi enkripsi algoritma IDEA dan bahasa yang digunakan adalah bahasa VHDL. Sebelumnya, penulis telah melakukan implementasi algoritma IDEA yang sesuai dengan pengimplementasian blok diagram algoritma tersebut tanpa adanya modifikasi. Sehingga hasil dari implementasi algoritma IDEA menjadi tidak efisien dapat dilihat dari gambar berikut.



Gambar 4. 1 Hasil Implementasi Standar Algoritma IDEA

Berdasarkan gambar diatas, dapat dilihat bahwa penggunaan *Slice* mencapai 162%, penggunaan *LUT* mencapai 156% dan penggunaan *IOB* mencapai 110%. Artinya *resource* yang dibutuhkan dalam desain sebelumnya tidak sesuai dengan kapasitas yang terdapat pada Xilinx Spartan 3-e. Hal-hal tersebut menyebabkan desain menjadi tidak optimal dan tidak dapat diimplementasikan pada FPGA Xilinx Spartan 3-e. Maka dari itu, penulis melakukan modifikasi pada desain algoritma IDEA berdasarkan perancangan yang telah dilakukan pada bab sebelumnya.

4.3.1 Implementasi Enkripsi

Pada proses implementasi enkripsi terdapat beberapa proses yaitu *Multiplexer*, *Basic Round*, *Final Round*, dan *Key Generator*. *Source code* pada penulisan ini akan ditampilkan pada lampiran tersendiri.

4.3.1.1 Multiplexer

Proses *multiplexer* digunakan untuk melakukan seleksi pada masukan 16-bit *plaintext* ataupun nilai pada 16-bit register dengan *control unit*, artinya ketika *control unit* memberikan sinyal S atau start telah bernilai 0 maka masukan dapat diproses serta ketika sinyal S atau start belum ada perubahan dari nilai awal (*default* S=1) maka keluaran tersebut akan di set 'X' karena masukan tidak dapat diproses. Proses *multiplexer* akan dijelaskan pada *pseudocode* berikut ini.

Tabel 4. 1 *Pseudocode Multiplexer*

Nama algoritma : Algoritma Enkripsi IDEA (Multiplexer)
Input : A0, A1 {A0 16-bit, A1 16-bit}
Output : Z {Z 16-bit}
Proses :

```
(S, A0, A1)
case Multiplexer is
    case 0 : set Z to A0;
    case 1 : set Z to A1;
    case others : set Z to "XXXXXXXXXXXXXXXX"
end case
```

Berdasarkan pseudocode tersebut, dijelaskan bahwa terdapat *input* A0 dan A1 sebagai dua pilihan yang akan masuk kedalam sistem dengan fungsional sesuai dengan *multiplexer*. Dengan keluaran dengan variabel Z, maka dapat diproses bahwa jika selektor akan memilih case nol maka variabel yang akan masuk ke dalam sistem adalah variabel A0 dan begitu juga jika selektor akan memilih case satu maka variabel yang akan masuk ke dalam sistem adalah variabel A1. Jika tidak keduanya, maka keluaran variabel Z akan *error* atau menghasilkan nilai X.

4.3.1.2 Register

Proses *register* digunakan sebagai blok penyimpanan sementara yang mana *register* tersebut akan menyimpan hasil dari *basic round* dan akan menjadi masukan kembali sesuai dengan jumlah round yang tersedia. Berdasarkan gambar 3.2 ketika jumlah *round* masih belum sama dengan delapan maka nilai keluaran dari *basic round* sementara akan disimpan di blok *register* dan ketika jumlah *round* pada proses tersebut sudah mencapai delapan maka nilai keluaran akan menjadi masukan pada *final round*, tidak akan disimpan kembali pada *register*. Proses *register* akan dijelaskan pada *pseudocode* berikut ini.

Tabel 4. 2 *Pseudocode Register*

```
Nama algoritma : Algoritma Enkripsi IDEA (Register)
Input : C0, enable
Output : B0
Proses :
(clk)
if rising_edge(clk)
    if enable to 1
        set C0 to B0;
    end if;
end if;
```

Berdasarkan pseudocode tersebut, dijelaskan bahwa variabel C0 sebagai masukan dan nantinya akan disimpan sementara sesuai dengan fungsional dari register itu sendiri. Dengan keluaran variabel B0 maka ketika terdapat sinyal clk

atau terdapat masukan yang akan disimpan maka variabel C0 sebagai penyimpan variabel dari B0.

4.3.1.3 Control Unit

Proses *control unit* merupakan salah satu elemen penting karena proses tersebut akan melakukan pengawasan terhadap masukan data atau informasi yang ada. Pada *control unit* yang dimaksud pada perancangan ini terdapat beberapa variabel pendukung yakni variabel *Ready, Starting* dan *Working*. Variabel-variabel tersebut dimaksudkan untuk melakukan pengawasan terhadap proses-proses yang ada pada algoritma IDEA seperti pada proses *multiplexer* yang menggunakan control unit sebagai selektor daripada *multiplexer* dan sebagainya. Proses *control unit* akan dijelaskan pada *pseudocode* berikut ini.

Tabel 4. 3 *Pseudocode Control Unit*

Nama algoritma : Algoritma Enkripsi IDEA (Control Unit)

Input : S, start, ready, enable, counterround sAA, sBB, sCC
Output : state
Proses :
(clk)
initialize states to (sAA, sBB, sCC)
if rising_edge(clk)
case based on state
when "0000" => state <= "0001";
when "0001" => state <= "0010";
when "0010" => state <= "0011";
when "0011" => state <= "0100";
when "0100" => state <= "0101";
when "0101" => state <= "0110";
when "0110" => state <= "0111";
when "0111" => state <= "1000";
when "1000" =>
if (start = '1') then
states := sBB;
else
states := sAA;
end if;
when others => state <= "1000";
states := sAA;
end case;

```
    case based on states
        when sAA => counterround <= state;
        ready <='1'; enable <='0';
        when sBB => counterround <= "0000";
        ready <= '0'; enable <= '1';
        S <= '0'; state <= "0001"; states := sCC;
        when sCC => counterround <= state;
        S <= '1';
    end case;
end if;
```

Berdasarkan pseudocode tersebut, dijelaskan bahwa control unit digunakan untuk mengawasi daripada variabel yang akan dipilih atau yang akan dijalankan berdasarkan variabel sAA, sBB dan sCC.

4.3.1.4 Key Generator

Proses *key generator* atau dapat dikatakan sebagai proses pembuatan kunci pada algoritma IDEA merupakan hal yang penting dalam alur enkripsi data. Pembuatan kunci maupun upa-kunci dilakukan untuk memenuhi konsep dasar pada algoritma IDEA yang mana dibutuhkan 52 *sub-keys* serta akan disebar pada seluruh *round*. 52 *sub-keys* tersebut dibutuhkan pada algoritma IDEA dan dibagi untuk delapan *round* pertama dengan pembagian enam sub-keys dan empat sub keys terakhir untuk *final round*. Proses *key generator* akan dijelaskan pada *pseudocode* berikut ini.

Tabel 4. 4 *Pseudocode Key Generator*

Nama algoritma : Algoritma Enkripsi IDEA (Key Generator)
Input : Round, Key {Round 4-bit, Key 128-bit}
Output : Out1, Out2, Out3, Out4, Out5, Out6 {Output 16-bit atau enam *subkeys* per round}
Proses :
(Round, Key)
//Pembagian 8 partisi dengan masing-masingnya 16-bit, kemudian dilakukan pergeseran kekiri 25-bit
Key_0 <=Key;
Key_1 <=Key_0(102downto0) & Key_0(127downto103);
Key_2 <=Key_1(102downto0) & Key_1(127downto103);
Key_3 <=Key_2(102downto0) & Key_2(127downto103);
Key_4 <=Key_3(102downto0) & Key_3(127downto103);
Key_5 <=Key_4(102downto0) & Key_4(127downto103);

```
Key_6 <=Key_5(102downto7)
Case Round is
    Case 0 : output <=Key_0(127downto32);
    Case 1 : output <=Key_0(31downto0) &key_1(127downto64);
    Case 2 : output <=Key_1(63downto0) &key_2(127downto96);
    Case 3 : output <=Key_2(95downto0);
    Case 4 : output <=Key_3(127downto32);
    Case 5 : output <=Key_3(31downto0) &key_4(127downto64);
    Case 6 : output <=Key_4(63downto0) &key_5(127downto96);
    Case 7 : output <=Key_5(95downto0);
    Case 8 : output <=Key_6;
End Case
Out1 <= output(95 downto 80);
Out2 <= output(79 downto 64);
Out3 <= output(63 downto 48);
Out4 <= output(47 downto 32);
Out5 <= output(31 downto 16);
Out6 <= output(15 downto 0);
```

Berdasarkan pseudocode tersebut, dijelaskan bahwa key generator akan melakukan pembagian terhadap kunci 128-bit awal yang akan dibagi menjadi delapan bagian dengan masing-masing 16-bit sehingga berdasarkan pada variabel Key_0 hingga Key_6 dilakukan pergeseran bit kunci ke kiri. Setelah itu dilakukan pembagian sesuai kebutuhan delapan round dan satu round akhir dapat dilihat pada pembagian case tersebut. Maka dari itu dilakukan inisialisasi akhir pada variabel Out1 hingga Out2 untuk menampung hasil akhir dari pembentukan kunci tersebut.

4.3.1.5 Basic Round

Proses *basic round* yang terdapat pada algoritma IDEA merupakan hal yang paling utama dari proses enkripsi algoritma tersebut. *Basic round* memiliki beberapa operasi aljabar yang berbeda, dimana operasi-operasi tersebut dilakukan untuk mendapatkan suatu keluaran enkripsi. Operasi-operasi yang terdapat pada basic round diataranya, operasi XOR, operasi penjumlahan modulo 2^{16}, dan operasi perkalian modulo (2^{16} + 1). Seluruh operasi tersebut dilakukan sebanyak delapan putaran. *Basic round* tersebut masing-masing putarannya menggunakan 6 upa-kunci, maka 48 dari 52 upa-kunci akan digunakan pada *basic round* tersebut. Proses *basic round* akan dijelaskan pada *pseudocode* berikut ini.

Tabel 4. 5 *Pseudocode Basic Round*

```
Nama algoritma : Algoritma Enkripsi IDEA (Basic Round)
Input :
I1,I2,I3,I4 {Plaintext 16-bit masing-masingnya}
Z1,Z2, Z3, Z4, Z5, Z6 {Key 16-bit masing-masingnya}
Output : O1,O2,O3,O4 {Temporary output 16-bit masing-masingnya}
Proses :
m1 : multiplier(I1,Z1);
a1 : adder(I2,Z2);
a2 : adder(I3,Z3);
m2 : multiplier(I4,Z4);
x1 : XOR(m1,a2);
x2 : XOR(a1,m2);
m3 : multiplier(x1,Z5);
a3 : adder(x2,m3);
m4 : multiplier(a3, Z6);
a4 : adder(m3,m4);
x3 : XOR(m1,m4);
x4 : XOR(a1,a4);
x5 : XOR(a2,m4);
x6 : XOR(m2,a4);
O1 <= x3;
O2 <= x5;
O3 <= x4;
O4 <= x6;
```

Berdasarkan pseudocode tersebut, dijelaskan bahwa penggunaan algoritma IDEA berdasarkan paten terdapat 14 proses yang didasari oleh tiga proses utama seperti fungsi multiplier, adder dan XOR yang akan melakukan sesuai dengan variabel masukannya yang kemudian disimpan pada variabel O1 hingga O4 sebagai variabel penyimpanan nilai sementara sebelum masuk ke proses final round.

4.3.1.6 Final Round

Proses *final round* yang terdapat pada algoritma IDEA merupakan kelanjutan daripada *basic round*. Maksudnya adalah ketika *basic round* telah melakukan delapan putaran maka tahap selanjutnya adalah proses *final round*. *Final round*

pada algoritma IDEA hanya dilakukan sekali proses atau dapat dikatakan tanpa perulangan proses, maka upa-kunci yang digunakan pada proses ini pun hanya dibutuhkan empat upa-kunci. Artinya setelah 48 upa-kunci yang telah digunakan pada *basic round*, maka bersisa empat upa-kunci terakhir untuk digunakan pada *final round* algoritma IDEA. Proses *final round* akan dijelaskan pada *pseudocode* berikut ini.

Tabel 4. 6 *Pseudocode Final Round*

```
Nama algoritma : Algoritma Enkripsi IDEA (Final Round)
Input :
O1,O2,O3,O4 {Temporary Output 16-bit masing-masingnya}
Z1,Z2, Z3, Z4 {Key 16-bit masing-masingnya}
Output : Y1,Y2,Y3,Y4 {Ciphertext 16-bit masing-masingnya}
Proses :
Y1 <= multiplier(O1,Z1);
Y2 <= adder(O3,Z2);
Y3 <= adder(O2,Z3);
Y4 <= multiplier(O4,Z4);
```

Berdasarkan pseudocode tersebut, dijelaskan bahwa proses yang dilakukan merupakan lanjutan dari proses sebelumnya dengan menggunakan variabel O1 hingga O4 sebagai masukan yang kemudian melakukan dua proses utama seperti multiplier dan adder. Dan diakhiri dengan hasil enkripsi algoritma IDEA pada variabel Y1 hingga Y4.

4.3.2 Implementasi *Input Output*

Pada proses implementasi *input output* terdapat beberapa subprogram yakni UART sebagai pemroses komunikasi serial antara Laptop/PC(Tx) dengan FPGA Xilinx Spartan 3-e(Rx) dan clkdiv sebagai pembagi clock serial yang mana digunakan untuk UART tersebut. *Source code* pada penulisan ini akan ditampilkan pada lampiran tersendiri.

4.3.2.1 UART

Pada subprogram UART berfungsi untuk melakukan komunikasi secara serial antara satu perangkat dengan perangkat lain. Dengan kata lain, komunikasi data yang dimulai dari Laptop/PC dan diterima oleh Laptop/PC, kemudian memulai proses enkripsi algoritma IDEA pada FPGA Xilinx Spartan 3-e dan akhirnya FPGA Xilinx Spartan 3-e akan mengirimkan kembali hasil enkripsi algoritma IDEA ke PC. Implementasi UART dapat dilihat pada *pseudocode* berikut ini.

Tabel 4. 7 *Pseudocode UART*

Nama algoritma : Algoritma Enkripsi IDEA (UART)

Input : reset, Read, Write, state, rx, recv,

Output : data

Proses :

```
(clk)
if clk'event and clk is 1
    if reset is 1
        set state to idle
        set Read to 1
        set Write to 1
    else
        if  state is idle
            set state to wait
        elsif state is wait
            set Write to 1
            if rx is 1
              set state to recv;
            end if
        elsif state is recv
            set Read to 0
            set state to read
        elsif state is read
            set x1 to data
            set x2 to data
            set x3 to data
            set x4 to data
            set Read to 1
            set state to wait_rx;
        elsif state is wait_rx
            if rx is 0
                set idea to 1
                set state to wait;
            end if
        elsif state is wait
            set state to write
```

```
        elsif state is write
            set data to y1
            set data to y2
            set data to y3
            set data to y4
            set state to write_rx
        end if
    end if
end if
```

Berdasarkan pseudocode tersebut, dijelaskan bahwa UART digunakan sebagai penghubung komunikasi antar perangkat. Dilakukan dengan beberapa kondisi yang akan mengirimkan variabel X1 hingga X4 sebagai variabel masukan pengguna kemudian dilakukan pula menerima variabel Y1 hingga Y4 sebagai hasil dari algoritma IDEA.

4.3.2.2 Clkdiv

Pada subprogram clkdiv merupakan pembagian sinyal *clock* yang mana pembagian tersebut akan digunakan pada proses UART atau berkomunikasi antar Laptop/PC dengan FPGA Xilinx Spartan 3-e. Implementasi clkdiv dapat dilihat pada pseudocode berikut ini.

Tabel 4. 8 *Pseudocode clkdiv*

```
Nama algoritma : Algoritma Enkripsi IDEA (clkdiv)
Input : clk
Output : clk_out
Proses :
(clk)
if clk'event and clk is 1
    counter++;
    if counter < 162
        set clk_out to 0
    elsif counter < 325
        set clk_out to 1
    else
        set counter to 0
    end if
end if
```

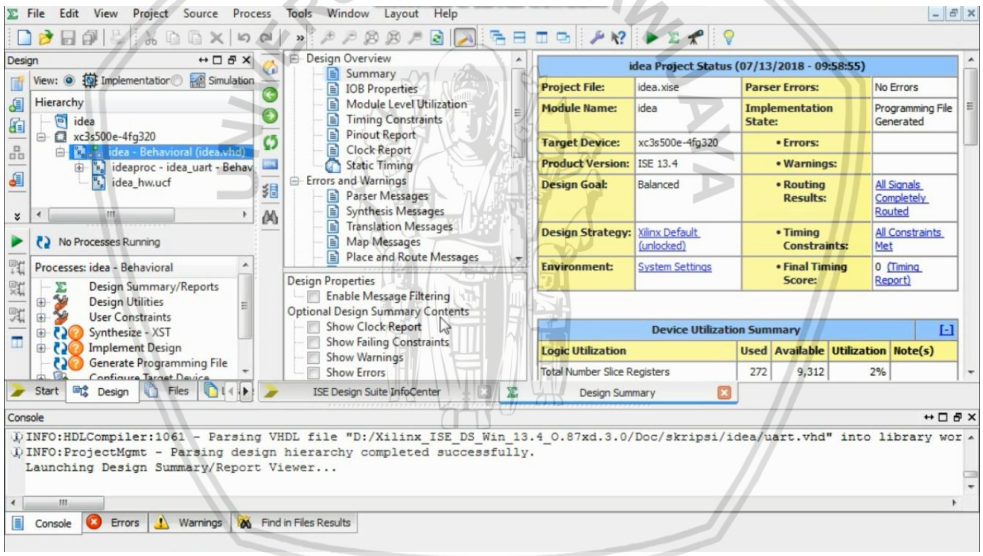
Berdasarkan pseudocode tersebut, dijelaskan bahwa dilakukan pembagian clock sesuai variabel baudrate yang digunakan. Baudrate yang digunakan pada penelitian ini adalah 9600 Hz, dari hal tersebut dilakukan pembagian seperti itu.

4.4 Implementasi Perangkat Keras

Implementasi enkripsi teks dengan algoritma IDEA pada FPGA berdasarkan perancangan sistem pada bab sebelumnya. Implementasi perangkat keras yaitu implementasi import ke FPGA dan setelah itu langsung dilakukan komunikasi serial antara Laptop/PC dengan FPGA sekaligus memberikan masukan melalui kolom *send sequences* dan menampilkan hasil keluaran berupa enkripsi dari algoritma IDEA.

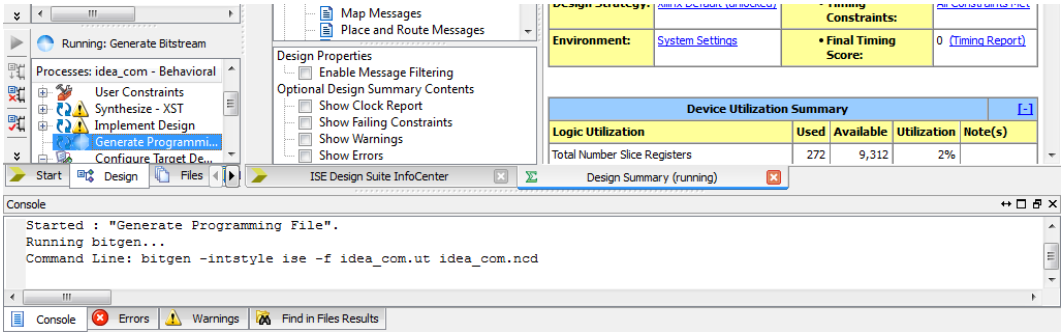
4.4.1 Implementasi *Import* Program ke FPGA

Tahap awal yang dilakukan sebelum import kedalam FPGA adalah memastikan bahwa program yang telah di desain kedalam ISE Project Navigator selesai, berhasil dijalankan dan tidak error dalam program. Proses *compile program* tersebut dapat dilihat sebagai berikut.



Gambar 4. 2 Proses *Compile Program*

Program pada *synthesize*, *translate, maps, place and route* dan *generate program* harus berhasil dan berwarna hijau atau berwarna kuning. Warna kuning menunjukkan dalam program masih terdapat beberapa peringatan, namun peringatan pada program tersebut tidak menggagalkan program. Artinya program dapat di import ke dalam FPGA. Tanda selesainya program dan dapat di import ke dalam FPGA sebagai berikut.

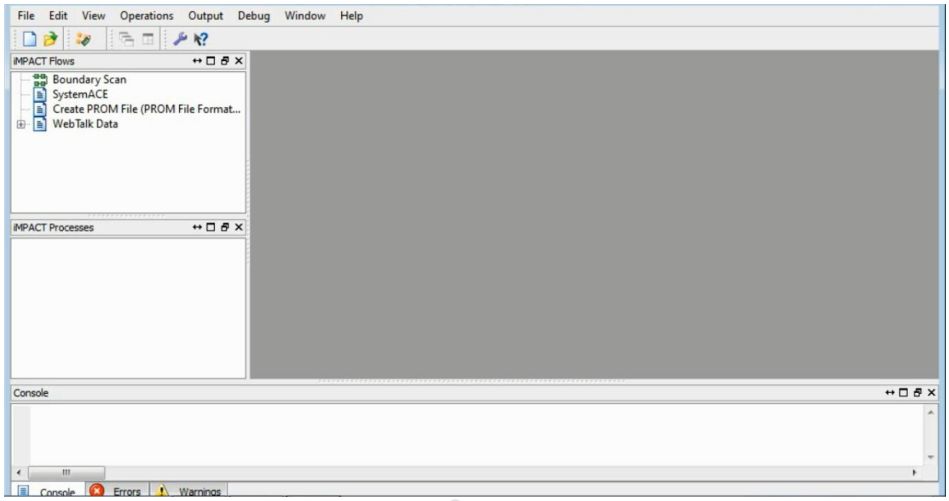


Gambar 4. 3 Program Sukses

Setelah itu, maka dilakukan proses import program ke dalam FPGA. Dalam hal ini dibutuhkan kabel USB untuk menghubungkan antara laptop dengan FPGA Xilinx Spartan 3-e. Berikut ini gambar dari implementasi tersebut.

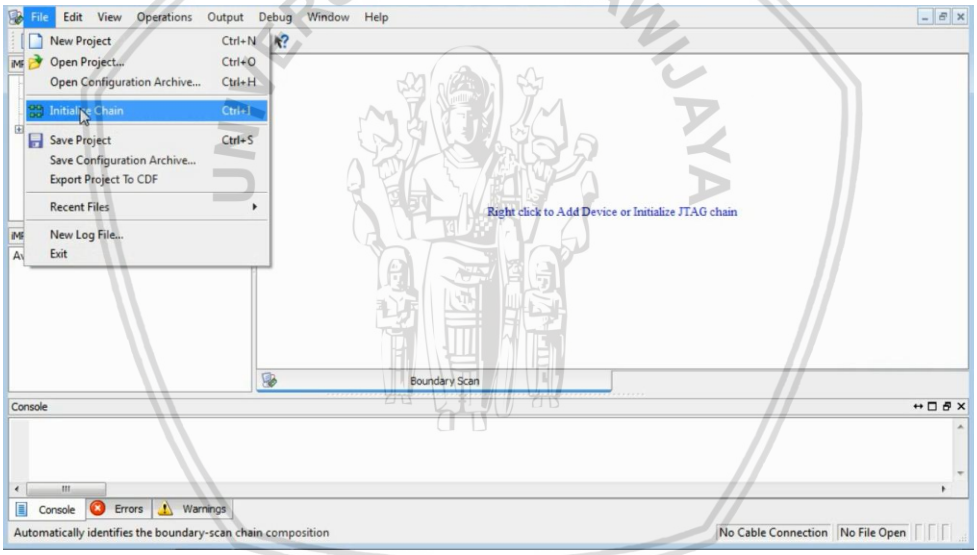
Gambar 4. 4 Menghubungkan Laptop dengan FPGA

Setelah laptop dan FPGA dihubungkan dengan kabel USB langkah selanjutnya adalah *configure target service* hal ini dilakukan untuk mengidentifikasi target import program yang dalam hal ini digunakan FPGA Xilinx Spartan 3-e. Pada saat klik *configure target service* akan keluar ISE impact pada modul ini akan digunakan untuk mengidentifikasi FPGA Xilinx Spartan 3-e. Berikut ini adalah gambar proses *configure target service*.



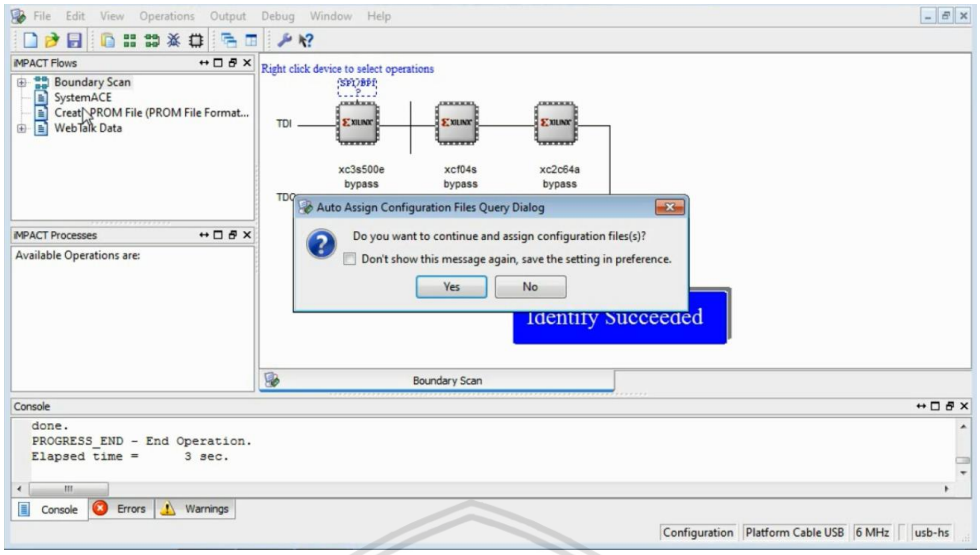
Gambar 4. 5 *Configure Target Service*

Langkah selanjutnya pilih *boundary scan* pada ISE *Impact*, setelah itu klik kanan pada ISE *Impact* dan pilih *Initialize Chain*. Hal ini dilakukan untuk mengidentifikasi IC yang akan digunakan seperti gambar berikut ini.



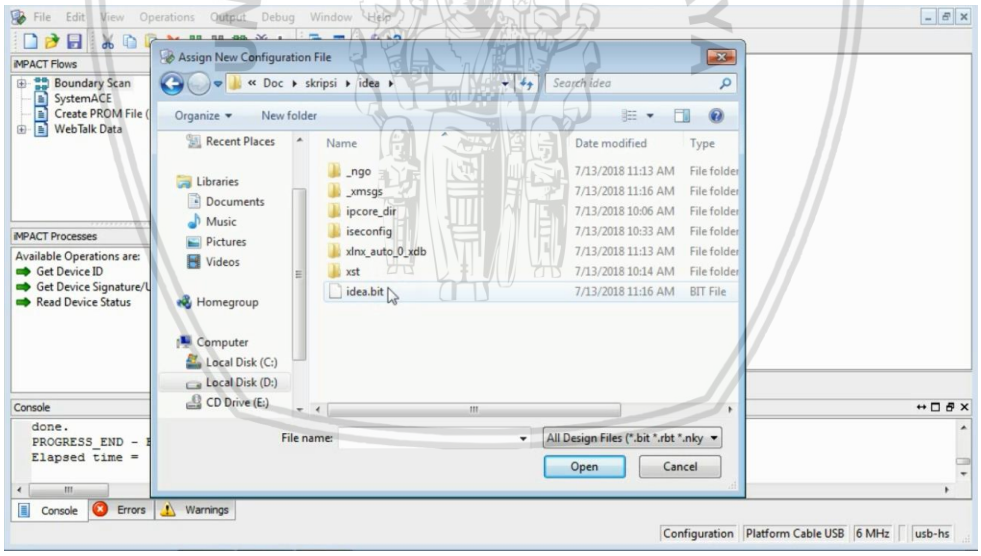
Gambar 4. 6 *Initialize Chain*

Setelah pilih initialize chain akan keluar pilihan IC yang akan digunakan dalam FPGA Xilinx Spartan 3-e dan pilihan yes untuk langkah selanjutnya seperti gambar berikut ini.



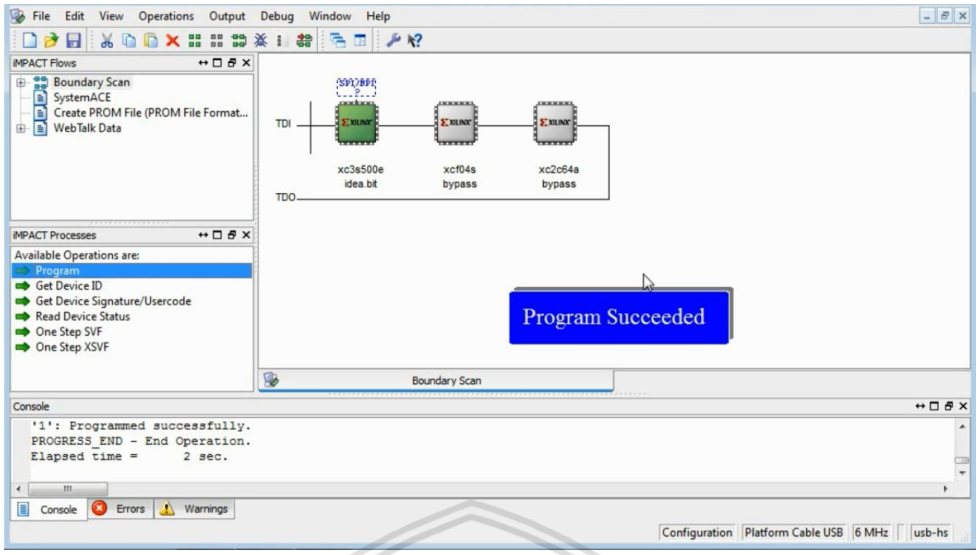
Gambar 4. 7 Identifikasi IC

Setelah pilih yes akan dengan sendirinya mengarahkan kepada IC xc3s500e yang akan digunakan untuk mengimport program. Langkah selanjutnya adalah pilih dimana file program disimpan, dalam penulisan ini program yang digunakan adalah idea.bit dan klik *open*. Proses pemilihan program dapat dilihat dari gambar berikut ini.



Gambar 4. 8 *Import* Program

Setelah program berhasil diimportkan pada IC xc3s500e akan berganti nama dengan nama program yang telah di importkan sebelumnya, yang mana dalam penulisan ini bernama idea.bit seperti gambar berikut ini.



Gambar 4. 9 *Import* Program Berhasil

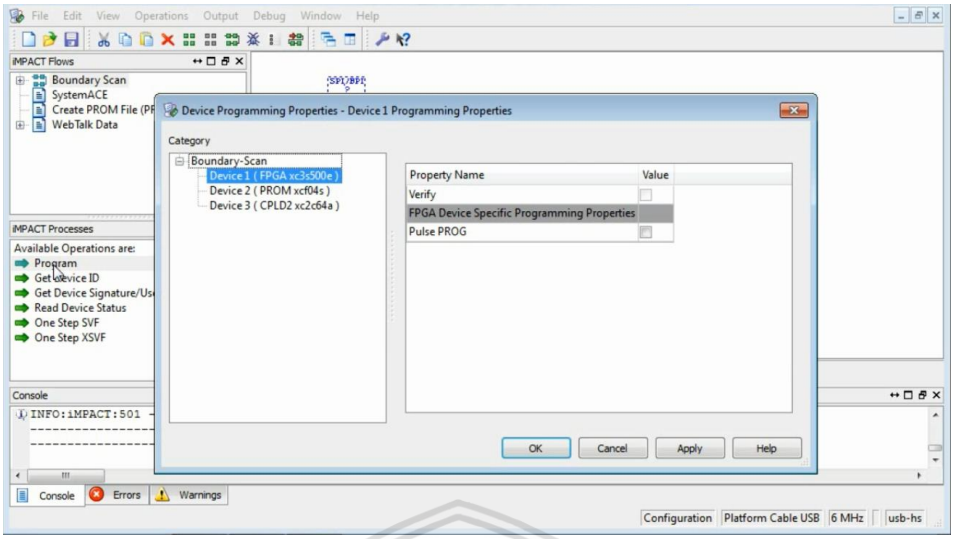
Pada langkah ini merupakan langkah terakhir dari proses *import program* dalam FPGA dan dalam proses ini program sudah berhasil di*import*kan di dalam FPGA.

4.4.2 Menampilkan Program

Langkah pertama adalah menghubungkan antara laptop dengan kabel USB dan menghubungkan kabel serial RS232 pada FPGA Xilinx Spartan 3-e seperti yang terdapat pada gambar berikut ini.

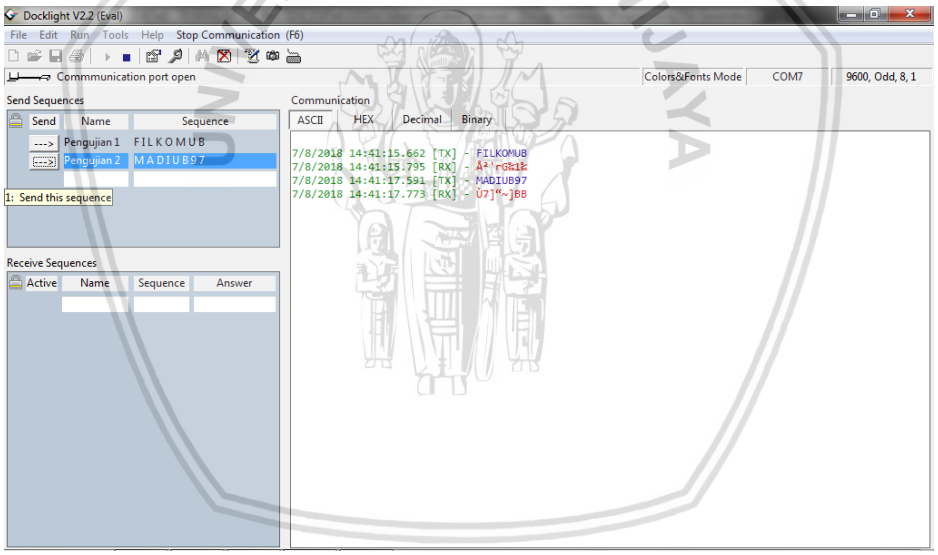
Gambar 4. 10 Menghubungkan Laptop, FPGA dan RS232

Setelah Laptop, FPGA dan RS232 berhasil dihubungkan langkah selanjutnya adalah klik kanan pada IC xc3s500e yang digunakan dan pilih program seperti yang terdapat pada gambar berikut ini.



Gambar 4. 11 Menjalankan Perintah Program

Setelah dilakukan proses pada gambar diatas, program akan menjalankan prosesnya dan menampilkannya pada aplikasi Docklight dengan bantuan kabel serial RS232 seperti yang terdapat pada gambar berikut ini.



Gambar 4. 12 Program Sukses dijalankan

Setelah dilakukan implementasi pada FPGA Xilinx Spartan 3-e dengan menggunakan algortima IDEA, didapatkan hasil implementasi berupa *ciphertext*, sehingga ketika dilakukan komunikasi pada PDA atau ponsel, data teks yang dikirimkan melalui proses komunikasi nirkabel antar PDA dapat dilakukan proses enkripsi terlebih dahulu. Dengan adanya proses enkripsi tersebut, maka data yang dikomunikasikan dapat terjamin kerahasiaan dan keutuhan datanya.

BAB 5 PENGUJIAN DAN ANALISIS

5.1 Skenario Pengujian

Skenario pengujian terhadap enkripsi data teks dengan algoritma IDEA pada FPGA berdasarkan pada bab sebelumnya. Pengujian dilakukan dengan tiga tahap yaitu pengujian fungsional, pengujian validitas dan pengujian waktu eksekusi. Pengujian yang pertama adalah pengujian fungsional. Pengujian ini dilakukan untuk mengetahui masukan dan hasil keluaran itu selaras. Pengujian ini dilakukan dengan cara membandingkan inputan dan output yang dikeluarkan.

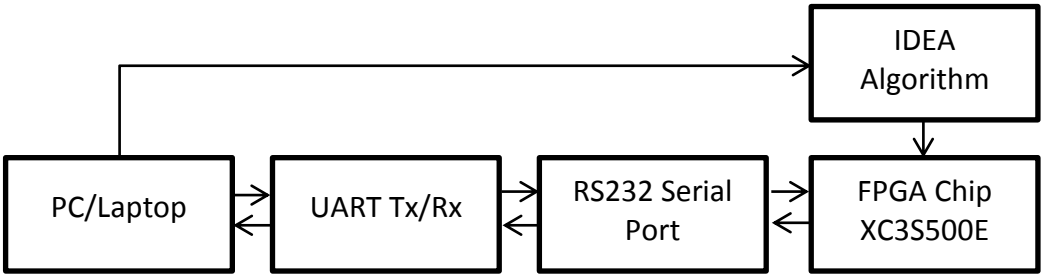
Pengujian kedua adalah pengujian validitas. Pengujian ini dilakukan dengan cara membandingkan hasil ciphertext yang terdapat pada referensi hasil dari vector test dengan kunci yang sama. Jika memang sistem memberikan hasil yang valid maka seharusnya hasil yang dikeluarkan akan sama dengan hasil yang terdapat pada referensi hasil dari vector test. Dari pengujian ini akan diambil 10 data yang akan di uji.

Pengujian ketiga adalah pengujian waktu eksekusi. Pengujian ini dilakukan untuk mengetahui waktu eksekusi yang dibutuhkan sistem untuk mengenkripsi data. Pengujian ini akan dilakukan dengan cara memasukkan beberapa data sampel yang berbeda. Hasil dari setiap data sampel akan dibandingkan dan di analisa.

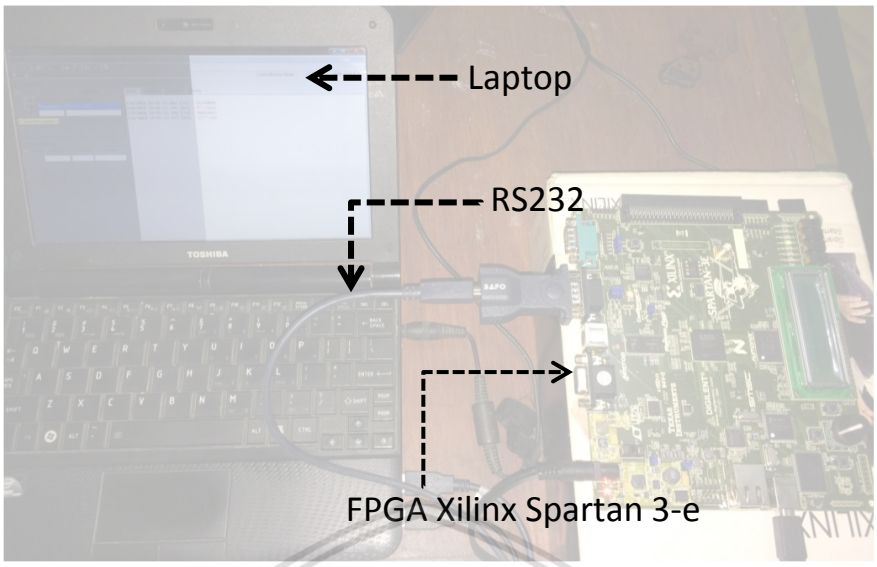
5.2 Hasil Pengujian

5.2.1 Pengujian Fungsional

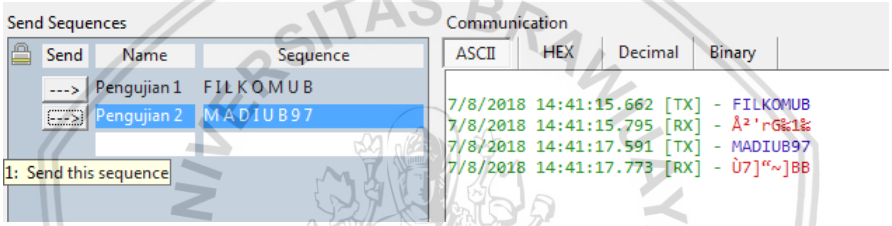
Pengujian fungsional dilakukan untuk mengetahui sistem dapat berjalan dengan baik berdasarkan perancangan yang ada. Pengujian tersebut dilakukan dengan cara menghubungkan seluruh rancangan sistem dan membandingkan masukan dan keluaran yang sesuai. Tujuan dari pengujian ini adalah untuk memastikan masukan dan keluaran yang dilakukan pengguna diterima dengan baik oleh sistem. Hasil dari pengujian ini akan dilakukan berdasarkan blok diagram berikut.



Gambar 5. 1 Blok diagram pengujian fungsional



Gambar 5. 2 Pengujian fungsional



Gambar 5. 3 Hasil masukan dan keluaran sistem

Dari hasil pengujian fungsional yang dilakukan didapatkan bahwa perancangan yang dirancang sebelumnya dapat di implementasikan dengan baik dan hasil seluruh masukan berupa beberapa tombol sesuai dengan yang diharapkan dan tidak terjadi satu kesalahan apapun.

5.2.2 Pengujian Validitas

Pengujian ini akan menggunakan 10 data acak yang hasilnya akan dibandingkan dengan beberapa referensi implementasi IDEA. Hasil dari pengujian ini dapat dilihat pada tabel berikut ini.

Tabel 5. 1 Hasil Pengujian Validitas

No	Plaintext	Ciphertext (ASCII)	Ciphertext (HEX)	Ket
1	MADIUB97	Ù7]“~]BB	D9375D937E5D4242	Benar
2	MADIWIJA	>}Åµä¦T	3E7DC51DB5E4A654	Benar
3	FILKOMUB	Å²'rG‰1‰	C5B2277247893189	Benar
4	MALANG?!	èCYÕS^À	E84359D553888DC0	Benar
5	UB1963TH	322êDâ$	333232EA44E2240D	Benar
6	madiub97	»jÙ ËÜ´µ	BB6AD90BCBDCB4B5	Benar

7	madiwija	#‰G\¨^é	2389475C1DA85EE9	Benar
8	filkomub	<*°øZ ê	3C2AB002F85AA0EA	Benar
9	malang?!	wÂ¶L¹@g	1A77C2B64CB94067	Benar
10	ub1963th	:?Á„	3A810E3FC1060484	Benar

Dari 10 data acak yang menjadi data uji tersebut mendapatkan hasil yang benar untuk keseluruhannya. Dari data diatas akan diambil satu data yang akan dibuktikan kebenarannya.

Plaintext (ASCII) = FILKOMUB

Plaintext (Hex) = 46494C4B4F4D5542

Ciphertext (Hex) = 3F92277247893189

Key (Hex) = 006400c8012c019001f4025802bc0320

R1 : Y = 4649 4C4B 4F4D 5542 ; SK = 0064 00c8 012c 0190 01f4 0258

R2 : Y = EAE2 CEF2 4096 3B1E ; SK = 02bc 0320 9002 5803 2003 e804

R3 : Y = FC36 13D8 E718 D802 ; SK = b005 7806 4000 c801 0640 07d0

R4 : Y = CAE5 B258 A450 90CF ; SK = 0960 0af0 0c80 0190 0320 04b0

R5 : Y = 3176 9AD4 3F64 C0A2 ; SK = a012 c015 e019 0003 2006 4009

R6 : Y = 2A7b 2E59 D385 C888 ; SK = 600c 800f 2bc0 3200 0640 0c80

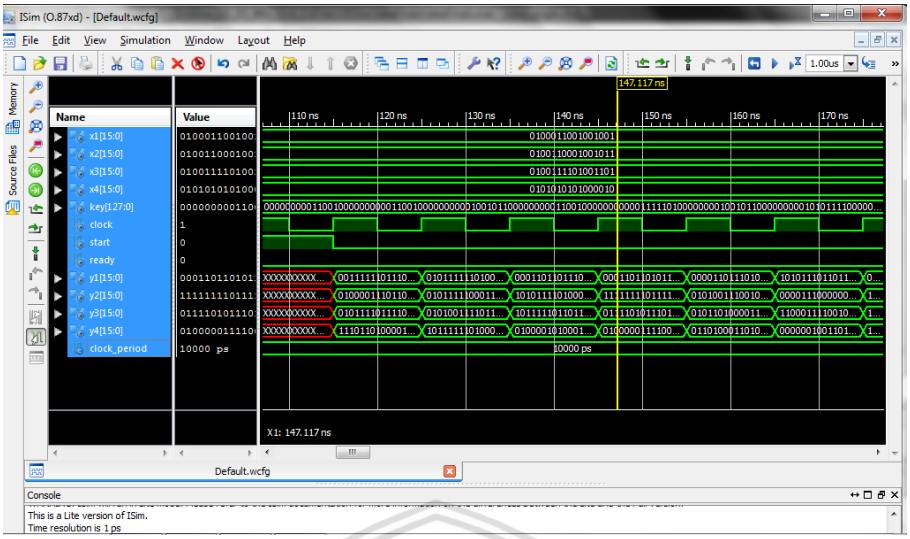
R7 : Y = 5AC2 A853 F502 33BC ; SK = 12c0 1900 1f40 2580 000c 8019

R8 : Y = 54B5 EE2C 78C1 BBFE ; SK = 0025 8032 003e 804b 0057 8064

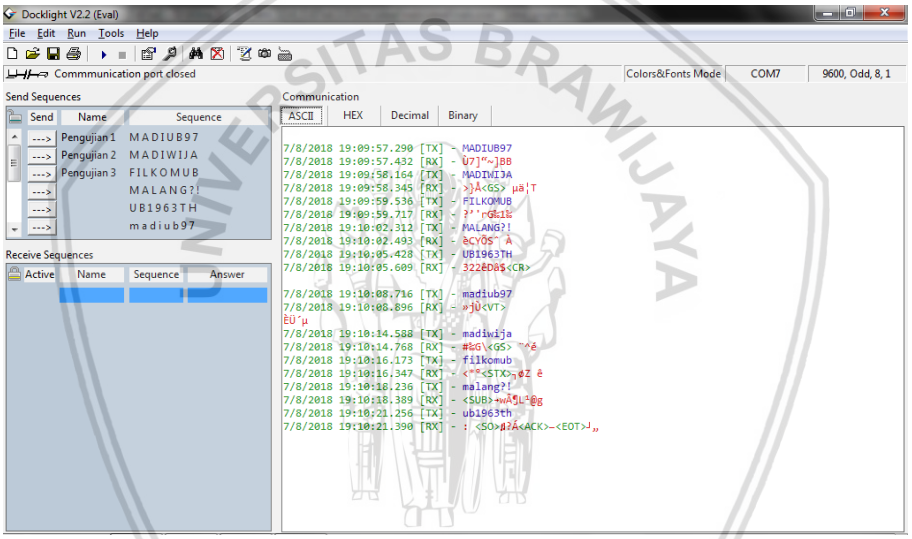
R9 : Y = A56B DC72 E389 0719 ; SK = 3200 4b00 6400 7d00 0000 0000

Out: Y = C5B2 2772 4789 3189

Dari *plaintext* FILKOMUB akan dijadikan dalam bentuk hex terlebih dahulu, setelah itu akan diproses dengan kunci yang telah tersedia. Dalam setiap *round* akan mendapatkan kunci yang baru berdasarkan hasil dari blok *key generator*. Dalam setiap round juga sudah dilakukan proses *basic round* sebanyak delapan putaran ditambah dengan *final round* sebanyak satu putaran dan pada akhirnya mendapatkan hasil "C5B2 2772 4789 3189" yang mana hasil tersebut dalam bentuk Hex. Berikut ini adalah gambar hasil dari enkripsi pada FPGA.



Gambar 5. 4 Hasil Simulasi Algoritma IDEA



Gambar 5. 5 Hasil Implementasi dengan Docklight

Dari pengujian validitas yang telah dilakukan maka dapat disimpulkan enkripsi yang berjalan pada FPGA berjalan dengan baik dan sesuai harapan.

5.2.3 Pengujian Waktu Eksekusi

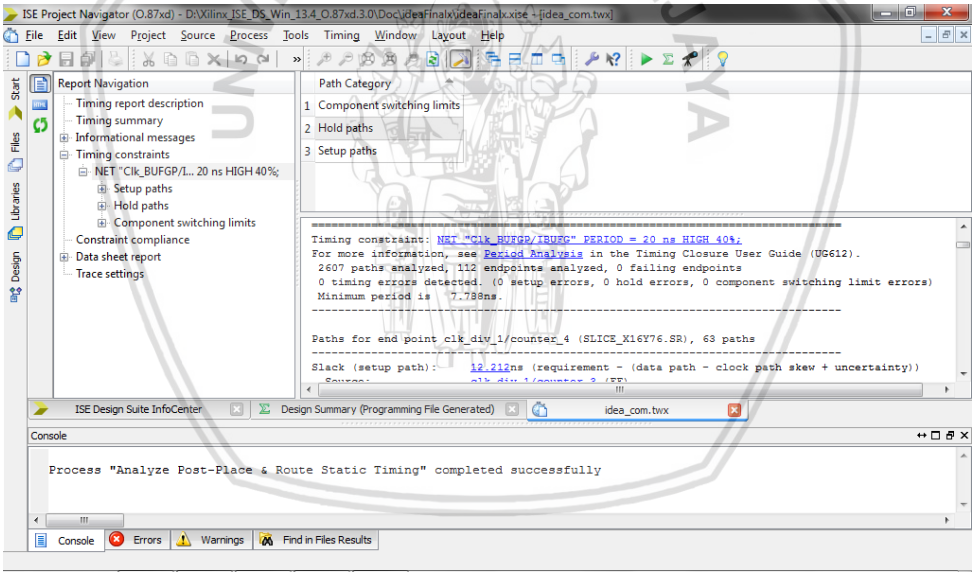
Pengujian waktu akan dilakukan dengan simulator pada ISE Project Navigator. Dari sana akan didapatkan waktu yang dibutuhkan. Dalam kasus ini dimasukkan 10 data yang sama dengan data yang dimasukkan pada pengujian validitas. Hasil dari pengujian waktu akan ditunjukkan pada tabel berikut ini.

Tabel 5. 2 Hasil pengujian waktu

No	Plaintext	Waktu
1	MADIUB97	7.788ns
2	MADIWIJA	7.788ns

3	FILKOMUB	7.788ns
4	MALANG?!	7.788ns
5	UB1963TH	7.788ns
6	madiub97	7.788ns
7	madiwija	7.788ns
8	filkomub	7.788ns
9	malang?!	7.788ns
10	ub1963th	7.788ns

Xilinx menyediakan *tools* untuk menampilkan hasil *timing summary* dari perancangan sistem. *Timing summary* akan diperoleh pada saat proses sintesis, hasil perhitungan daripada *software* xilinx. Hasil *timing summary* merupakan hasil perhitungan secara simulasi sehingga tidak dapat menggambarkan *latency time*. Perancangan sistem enkripsi algoritma IDEA pada FPGA Xilinx Spartan 3-e menghasilkan *timing* summary sebagai berikut.



Gambar 5. 6 *Analyze Post-Place & Route Static Timing*

Berdasarkan gambar tersebut dilakukan pengujian dengan memasukkan data 64-bit pada program. Pada proses yang berjalan berdasarkan *timing analyzer* ISE Project Navigator program algoritma IDEA yang didesain dapat melakukan enkripsi dalam waktu minimum 7.788ns yang artinya dari hasil pengujian ini mendapatkan data bahwa bagaimanapun karakter pada enkripsi algoritma IDEA tidak mempengaruhi waktu yang dibutuhkan selama panjang karakter tersebut masih dalam satu blok yaitu 64-bit.

5.3 Hasil Analisis

5.3.1 Analisis Fungsional

Berdasarkan pengujian fungsional yang dilakukan untuk mengetahui sistem yang dapat berjalan dengan baik berdasarkan perancangan yang ada. Pengujian tersebut dilakukan dengan cara menghubungkan seluruh rancangan sistem dan membandingkan masukan dan keluaran yang sesuai. Tujuan dari pengujian ini adalah untuk memastikan masukan dan keluaran yang dilakukan pengguna diterima dengan baik oleh sistem.

Hasil pada pengujian fungsional didapatkan bahwa perancangan yang dirancang sebelumnya dapat di implementasikan dengan baik dan hasil seluruh masukan berupa beberapa tombol sesuai dengan yang diharapkan dan tidak terjadi satu kesalahan apapun. Hal ini dibuktikan dengan perancangan pada *Power Socket*, RS232, FPGA *Board* dan PC masing-masingnya berjalan dengan baik. Implementasi sistem enkripsi algoritma IDEA pada FPGA Xilinx Spartan 3-e XC3S500E menghasilkan *report* implementasi yang terlihat pada tabel berikut ini.

Tabel 5. 3 Penggunaan logika pada implementasi algoritma IDEA

Jenis Pemakaian Logika	Kapasitas yang digunakan	Kapasitas yang tersedia	Persentase Pemakaian
Jumlah *Slices*	575	4656	12%
Jumlah LUT	879	9312	9%
Jumlah IOB	12	232	5%
Jumlah MULTI18x18	6	20	30%

Penggunaan kapasitas logika dapat diminimalkan dengan meminimalkan perancangan sistem serta tata cara pemrograman VHDL. Berdasarkan hasil yang terdapat pada tabel diatas, disimpulkan bahwa pada sistem yang dilakukan pada penelitian ini lebih optimal dan dapat di implementasikan pada FPGA karena pemakaian kapasitas logikanya tidak melebihi kapasitas logika yang tersedia pada FPGA.

5.3.2 Analisis Validitas

Berdasarkan pengujian validitas yang menggunakan 10 data acak yangmana hasilnya akan dibandingkan dengan beberapa referensi implementasi IDEA. Dari pengujian tersebut diambil sebuah sampel *plaintext* FILKOMUB akan dijadikan dalam bentuk hex terlebih dahulu, setelah itu akan diproses dengan kunci yang telah tersedia. Dalam setiap *round* akan mendapatkan kunci yang baru berdasarkan hasil dari blok *key generator*. Dalam setiap *round* juga sudah dilakukan proses *basic round* sebanyak delapan putaran ditambah dengan *final round* sebanyak satu putaran dan pada akhirnya mendapatkan hasil “C5B2 2772

4789 3189” yang mana hasil tersebut dalam bentuk Hex. Dari pengujian validitas yang telah dilakukan maka dapat disimpulkan enkripsi yang berjalan pada FPGA berjalan dengan baik dan sesuai harapan.

5.3.3 Analisis Waktu Eksekusi

Berdasarkan pengujian waktu yang dilakukan dengan simulator pada ISE Project Navigator, didapatkan waktu yang dibutuhkan untuk melakukan proses enkripsi algoritma IDEA pada FPGA Xilinx Spartan 3-e. Dalam kasus ini dimasukkan 10 data yang sama dengan data yang dimasukkan pada pengujian validitas. Pengujian tersebut dilakukan dengan memasukkan data 64-bit pada program.

Hasil waktu eksekusi pada penelitian ini dilakukan perbandingan dengan penelitian yang ditulis oleh Allen Michalski, Kris Gaj dan Tarek El-Ghazawi pada tahun 2003. Penelitian tersebut menghasilkan waktu eksekusi algoritma IDEA sebesar 12.191ns. Pada proses yang berjalan berdasarkan *timing analyzer* ISE Project Navigator program algoritma IDEA yang didesain pada penelitian ini dapat melakukan enkripsi dalam waktu minimum 7.788ns yang artinya dari hasil perbandingan dan pengujian disimpulkan bahwa penelitian ini lebih efisien untuk diterapkan pada perangkat yang membutuhkan proses respon yang cepat dan mendapatkan data bahwa bagaimanapun karakter pada enkripsi algoritma IDEA tidak mempengaruhi waktu yang dibutuhkan selama panjang karakter tersebut masih dalam satu blok yaitu 64-bit.

BAB 6 PENUTUP

6.1 Kesimpulan

Berdasarkan hasil perancangan, implementasi dan pengujian yang dilakukan maka diambil kesimpulan sebagai berikut.

1. Perancangan enkripsi data text dengan algoritma *International Data Encryption Algorithm* pada perangkat kriptografi berbasis FPGA dapat dilakukan dengan memasukkan data *entry keyboard* & *display input* kemudian diproses menggunakan *serial communication* rs232 dan mengirimkan data ke FPGA. Selanjutnya dilakukan proses enkripsi data dengan algoritma IDEA sehingga didapatkan keluaran *ciphertext*.
2. Implementasi enkripsi data teks di *embedded system* dengan metode *International Data Encryption Algorithm* dapat diimplementasikan pada FPGA Xilinx Spartan 3-e. Dengan FPGA Xilinx Spartan 3-e yang digunakan sebagai *simulator pemrograman hardware* dapat berhasil dilakukan, maka ketika dilakukannya pembentukan pada IC kriptoprosesor dapat meminimalisir kesalahan.
3. Berdasarkan hasil pengujian yang dilakukan, pada pengujian pertama yaitu pengujian fungsional melalui perancangan yang tersedia didapatkan bahwa sistem dapat diimplementasikan dengan baik dan hasil seluruh fungsional sesuai dengan yang diharapkan dan tidak terjadi satu kesalahan pun. Pada pengujian kedua yaitu pengujian validitas, menghasilkan keluaran yang sesuai harapan. Maka status validitasnya dikatakan *valid* dan pengujian validitas memberikan hasil 100% sistem benar dalam menghasilkan enkripsi suatu data teks. Berdasarkan hasil pengujian waktu, perbedaan karakter tidak mempengaruhi waktu yang digunakan untuk mengenkripsi data teks selama panjang karakter tersebut tidak melebihi dari satu panjang blok yaitu delapan karakter atau 64-bit. Pengujian waktu memberikan hasil waktu yang diperlukan untuk mengenkripsi data teks di FPGA Xilinx Spartan 3-e adalah 7.788ns.

6.2 Saran

Saran yang dapat diberikan untuk pengembangan penelitian ini antara lain:

1. Untuk pengembangan lebih lanjut, diharapkan diciptakan juga untuk dekripsi data teks.
2. Untuk pengembangan lebih lanjut, diharapkan telah dapat diimplementasi kan di dalam IC kriptoprosesor yang sebenarnya sehingga enkripsi berbasis *embedded system* dapat segera tercipta.
3. Untuk pengujian lebih lanjut, bisa dilakukan pengujian serangan seperti *exhaustive attack* atau *brute force*.

DAFTAR PUSTAKA

Modugu, Rajashekhar., Kim, Yong-Bin., dan Minsu Choi. 2012. *Design and Performance Measurement of Efficient IDEA (International Data Encryption Algorithm) Crypto-Hardware using Novel Modular Arithmetic Components*. Boston: Journal Science and Technology.

Al-Haija, Qasem Abu., Smadi, Mahmoud., Al-Ja'fari , Monther., dan Al-Shua'ibi, Abdullah. 2014. *Efficient FPGA Implementation of RSA Coprocessor Using Scalable Modules*. Al-Ahsa: International Symposium on Emerging Inter-networks, Communication and Mobility.

Syahral, Mohamad. 2012. *Perancangan dan Implementasi Algoritma Arcfour pada Perangkat Kriptografi berbasis FPGA*. Depok.

Munir, Rinaldi. 2016. *Bahan Kuliah IF5054 Kriptografi.* Departemen Teknik Informatika, Institut Teknologi Bandung. Jatinangor.

Chang, H.-S., 2004. *International Data Encryption Algorithm.*

Ms Snehal Patil, P. V. B. D. o. C. B. P., 2014. Encryption Algorithm for Increasing Security. *International Journal of Application od Innovation in Engineering J Management (IJAIEM),* 3(8).

Osama Almasri, H. M. J., 2013. Introducing an Encryption Algorithm based on IDEA. *International Journal of Science and Research (IJSR), India Online ISSN : 2319 -7064,* 2(0).

Attila Strba, R. &. (2009). Embedded systems with limited power resources . *WHITE PAPER*, 1.

Munir, R. (2004). *Pengantar Kriptografi.* Bandung: Departemen Teknik Informatika ITB.

Osama Almasri, H. M. (2013). Introducing an Encryption Algorithm based on IDEA. *International Journal of Science and Research (IJSR)*, 334.

Prasanna, A. D. (2005). Configuration Compression for FPGA-Based Embedded Systems. *IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS*, 1394.

Tolstrup, T. K., Nielson, H. R., & Nielson, F. (2007). *Language-based Security for VHDL.* Denmark: Technical University of Denmark.

Dorottya Papp, Z. M. (2015). Embedded Systems Security: Threats, Vulnerabilities, and Attack Taxonomy. *2015 Thirteenth Annual Conference on Privacy, Security and Trust (PST)*, 145.

E. Wanderley, R. V.-P.-L. (2012). Security FPGA Analysis. 10-12.

Massey, L. (1993). *Paten No. 5214703.* U.S.

Suprapti, I. (2003). *STUDI SISTEM KEAMANAN DATA DENGAN METODE PUBLIC KEY CRYPTOGRAPHY.* Bandung: Institut Teknologi Bandung.

Ian Kuon, R. T. (2008). FPGA Architecture: Survey and Challenges. *Foundations and Trends in Electronic Design Automation* , 135-253.

Mehrdad Majzoobi, F. K. (2010). FPGA-Oriented Security. California: Electrical and Computer Engineering Department, Rice University.

Xilinx. (2013). *Spartan3-E FPGA Family Data Sheet.* United States: Xilinx.